

版权相关注意事项:

- 1、书籍版权归著者和出版社所有
- 2、本PDF来自于各个广泛的信息平台，经过整理而成
- 3、本PDF仅限用于非商业用途或者个人交流研究学习使用
- 4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
- 5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
- 6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
- 7、请于下载PDF后24小时内研究使用并删掉本PDF

版权相关注意事项:

- 1、书籍版权归著者和出版社所有
- 2、本PDF来自于各个广泛的信息平台，经过整理而成
- 3、本PDF仅限用于非商业用途或者个人交流研究学习使用
- 4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
- 5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
- 6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
- 7、请于下载PDF后24小时内研究使用并删掉本PDF



基于实战开发，逐步详解网站代码  
借助生活物品，快速理解复杂知识  
引导问题思考，轻松解决技术难题

利用百余案例，加强代码实战能力  
通过丰富习题，巩固加深知识理解  
遵循开发标准，编写优质程序代码

清华大学出版社

版权相关注意事项:

- 1、书籍版权归著者和出版社所有
- 2、本PDF来自于各个广泛的信息平台，经过整理而成
- 3、本PDF仅限用于非商业用途或者个人交流研究学习使用
- 4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
- 5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
- 6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
- 7、请于下载PDF后24小时内研究使用并删掉本PDF



本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目(CIP)数据

HTML5 布局之路/刘国利编著. —北京：清华大学出版社，2017(2018.9重印)  
ISBN 978-7-302-46684-0

I. ①刘… II. ①刘… III. ①超文本标记语言—程序设计 IV. ①TP312.8

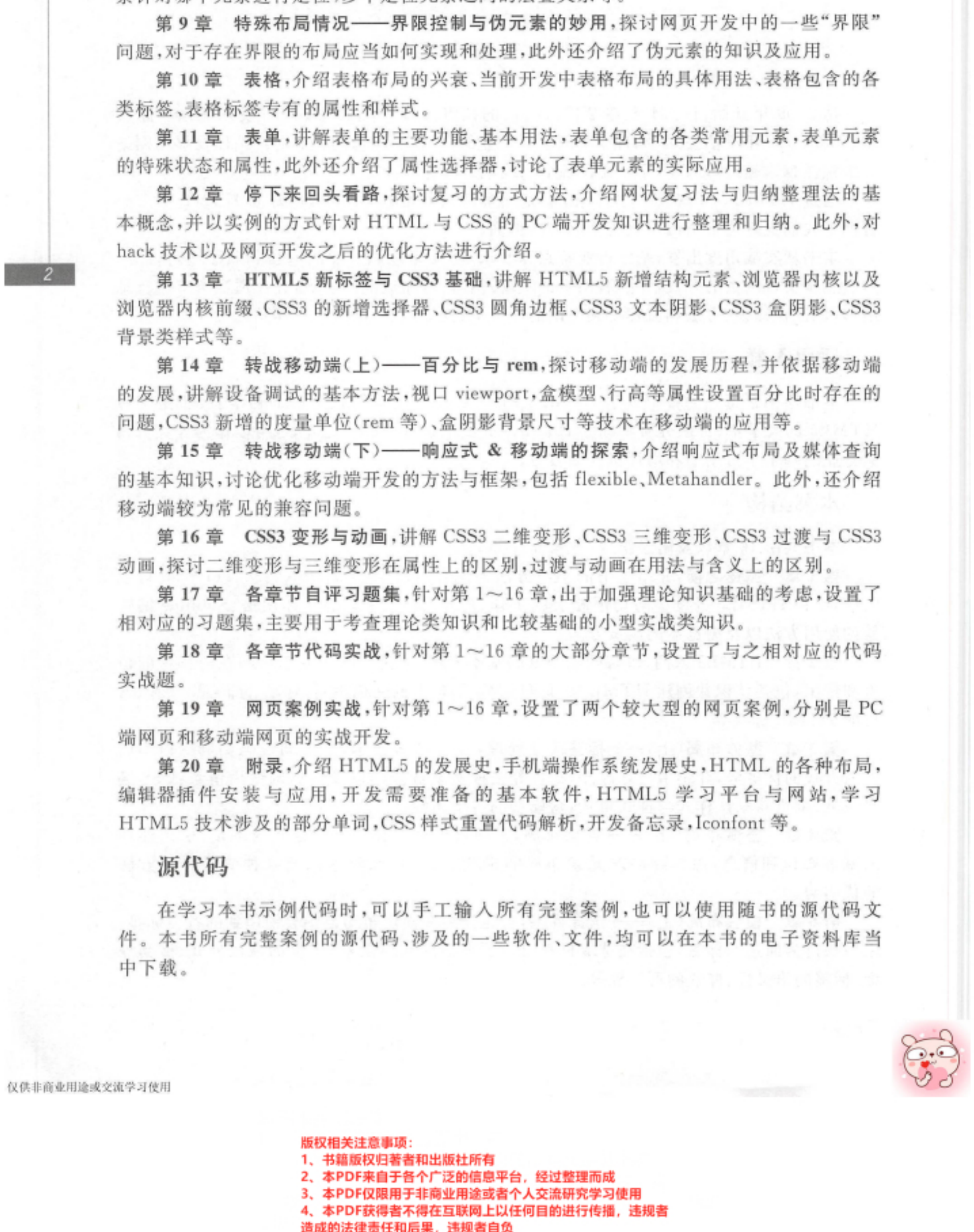
中国版本图书馆 CIP 数据核字(2017)第 035908 号

责任编辑：贾 斌 薛 阳  
封面设计：何凤霞  
责任校对：时翠兰  
责任印制：李红英

出版发行：清华大学出版社  
网 址：http://www.tup.com.cn, http://www.wqbook.com  
地 址：北京清华大学学研大厦 A 座 邮 编：100084  
社 总 机：010-62770175 邮 购：010-62786544  
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn  
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn  
课件下载：http://www.tup.com.cn, 010-62795954

印 装 者：三河市铭诚印务有限公司  
经 销：全国新华书店  
开 本：190mm×260mm 印 张：36 字 数：878 千字  
版 次：2017 年 6 月第 1 版 印 次：2018 年 9 月第 2 次印刷  
印 数：3001~3500  
定 价：89.00 元

产品编号：071571-01



版权相关注意事项:

- 1、书籍版权归著者和出版社所有
- 2、本PDF来自于各个广泛的信息平台，经过整理而成
- 3、本PDF仅限用于非商业用途或者个人交流研究学习使用
- 4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
- 5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
- 6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
- 7、请于下载PDF后24小时内研究使用并删掉本PDF





# 目录

## CONTENTS

第 1 章 旅途之前	1
1.1 讲解方法与旅途指南	1
1.1.1 和其他技术书籍相比,本书到底有何不同	1
1.1.2 书籍概览	1
1.1.3 推荐学习流程	3
1.1.4 旅途指南	3
1.2 HTML5 学习的前期准备	3
1.3 认识 HTML5	4
1.3.1 HTML5 是做什么的	4
1.3.2 HTML5 的由来	4
1.3.3 HTML5 的基本组成	4
1.3.4 专业化角度认识 HTML5	4
1.3.5 容易混淆的概念	5
1.4 了解 HTML5 行业前景与基本概念	6
1.4.1 一些行业词汇	6
1.4.2 HTML5 的行业并非一劳永逸	6
1.4.3 近几年来 HTML5 行业的变化	7
1.4.4 到底是什么决定着薪资	7
1.4.5 是谁决定你的去留	8
1.5 HTML 与 CSS 的学法	8
1.5.1 方法 1 整体到局部,骨架到血肉	8
1.5.2 方法 2 类比	8
1.5.3 方法 3 记忆很重要	8
1.5.4 方法 4 聚沙成塔	9
1.5.5 方法 5 循序渐进	9
1.5.6 方法 6 知识的迁移	9
1.5.7 方法 7 生活辅助学习	9
1.5.8 方法 8 实践出真知	10
1.6 开发工程师与 Photoshop	10







1.6.1	图片切图 .....	10
1.6.2	认识软件 .....	10
1.7	切图与 Photoshop 相关用法 .....	11
1.7.1	切图的基本流程 .....	11
1.7.2	打开文件 .....	11
1.7.3	找到切图目标 .....	12
1.7.4	整理好要处理的图层 .....	14
1.7.5	新建文件存储目标图像 .....	15
1.7.6	在新文件中调整图像位置 .....	17
1.7.7	修改画布大小 .....	18
1.7.8	将图片按照最佳格式类型进行存储 .....	19
1.7.9	图片大小处理与压缩 .....	23
1.8	使用 Photoshop 获取图层信息 .....	24
1.8.1	获取颜色 .....	24
1.8.2	文字内容与特点 .....	26
1.8.3	撤销 .....	26
1.8.4	圆角大小的测量 .....	26
1.8.5	阴影的测量 .....	28
1.8.6	将文字处理成图像 .....	29
1.8.7	Photoshop 快捷键总结 .....	29
1.9	代码编辑器 .....	29
1.9.1	Sublime Text .....	29
1.9.2	其他代码编辑器 .....	31
1.10	浏览器调试 .....	32
1.10.1	为何要进行浏览器调试 .....	32
1.10.2	浏览器调试的基本要求 .....	32
1.10.3	浏览器调试方法 .....	33
第2章	HTML5 入门 .....	34
2.1	网站开发流程 .....	34
2.1.1	网站开发流程图 .....	34
2.1.2	网站需求调查阶段 .....	34
2.1.3	网站技术分析阶段 .....	36
2.1.4	网站页面策划阶段 .....	36
2.1.5	网站设计开发阶段 .....	36
2.1.6	网站测试改进阶段 .....	37
2.1.7	前端工程师负责的部分 .....	37
2.2	第一个 HTML 文件 .....	37
2.2.1	创建基本的网站文件夹 .....	37







2.2.2	创建第一个 HTML 文件 .....	38
2.2.3	HTML 标签的书写规则 .....	39
2.2.4	HTML 书写规则的问题区 .....	40
2.3	基本的 HTML 结构 .....	42
2.3.1	一个 HTML 文件的基本组成 .....	42
2.3.2	文档声明 .....	42
2.3.3	title 标题 .....	43
2.3.4	meta 元信息 .....	43
2.3.5	HTML 文件的内容区 .....	46
2.3.6	HTML 注释 .....	46
2.3.7	网站开发常用标签 .....	47
2.3.8	基本 HTML 结构的问题区 .....	50
第 3 章	整体布局(上)——标签尺寸处理 .....	53
3.1	整体布局与整体布局中使用的标签 .....	53
3.1.1	整体布局 .....	53
3.1.2	div 元素 .....	54
3.2	什么是 CSS .....	55
3.2.1	没有 CSS 时代的网页 .....	55
3.2.2	什么是 CSS .....	57
3.3	CSS 引入方式 .....	57
3.3.1	行内书写——最简单的样式操作方法 .....	57
3.3.2	内部书写——简化样式操作 .....	59
3.3.3	外部引入——控制多页面样式 .....	60
3.3.4	CSS 三种常见引入方式比较 .....	63
3.3.5	外部引入 CSS 的扩展知识 .....	65
3.3.6	CSS 引入方式的问题区 .....	66
3.4	CSS 选择器 .....	68
3.4.1	生活中的“选择器”——找人 .....	68
3.4.2	CSS 选择器的基本语法 .....	69
3.4.3	CSS 基本选择器 .....	69
3.4.4	样式冲突的问题 .....	72
3.4.5	生活中的“优先级”——谁是老大 .....	74
3.4.6	CSS 选择器优先级 .....	74
3.4.7	行内的 style 属性 .....	77
3.4.8	选择器的使用原则 .....	77
3.4.9	CSS 选择器的问题区 .....	78
3.5	CSS 编码规范 .....	78
3.5.1	CSS 注释 .....	78







3.5.2	书写风格 .....	79
3.5.3	关于类名命名 .....	80
3.5.4	样式书写顺序 .....	80
3.5.5	CSS 编码规范的问题区 .....	81
3.6	CSS 样式重置 .....	83
3.6.1	什么是样式重置 .....	83
3.6.2	为何需要样式重置 .....	84
3.6.3	样式重置文件 .....	85
3.7	盒模型 .....	86
3.7.1	生活中的“盒模型”——鱼缸 .....	86
3.7.2	盒模型基本知识 .....	86
3.7.3	盒模型——width 与 height 属性 .....	87
3.7.4	盒模型——margin 属性 .....	88
3.7.5	盒模型——padding 属性 .....	90
3.7.6	盒模型——border 属性 .....	91
3.7.7	盒模型的问题区 .....	94
第 4 章	整体布局(下)——浮动布局 .....	103
4.1	浮动 .....	103
4.1.1	为何要浮动 .....	103
4.1.2	生活中的“浮动”——水槽 .....	103
4.1.3	浮动——float 属性 .....	103
4.1.4	浮动特效分析 .....	104
4.1.5	浮动的问题区 .....	112
4.2	浮动的影响 .....	113
4.2.1	文档流 .....	113
4.2.2	浮动元素对父级元素高度的影响 .....	113
4.2.3	浮动元素对兄弟级元素布局的影响 .....	116
4.3	清除浮动 .....	117
4.3.1	浮动——clear 属性 .....	117
4.3.2	清除浮动的不同类型 .....	118
4.3.3	为兄弟元素设置 clear 样式 .....	118
4.3.4	空标签清除浮动 .....	120
4.3.5	br 标签清除浮动 .....	120
4.3.6	父元素浮动 .....	121
4.3.7	父元素设置 overflow: hidden 或 auto .....	122
4.3.8	利用 after 伪元素清除浮动 .....	122
4.3.9	after 伪元素清除浮动的实际用法 .....	124
4.4	关于“清除浮动”的问题区 .....	125





4.4.1	clear: both 的兼容问题 .....	125
4.4.2	为父级设置高度是不是清除浮动的方法 .....	125
4.4.3	浮动元素与非浮动元素处于同一行时会出现的问题 .....	125
4.4.4	清除浮动方法的选择 .....	126
4.5	较为复杂的浮动布局 .....	126
4.5.1	功能需求 .....	126
4.5.2	需求分析 .....	127
第 5 章	模块布局(上)——选择标签 .....	131
5.1	为何要选择标签 .....	131
5.2	开发时可选用的标签以及功能 .....	131
5.2.1	h1~h6 标题类标签 .....	131
5.2.2	hr 分隔线 .....	132
5.2.3	p 与 br 段落与换行 .....	132
5.2.4	无序列表与有序列表 .....	132
5.2.5	自定义列表 .....	133
5.2.6	行内标签 .....	133
5.2.7	代码范例的显示效果图 .....	134
5.3	标签选择时的影响因素 .....	135
5.4	标签的默认显示样式 .....	135
5.4.1	显示属性 display .....	135
5.4.2	根据标签默认 display 属性划分类别 .....	136
5.4.3	显示样式影响的标签选用 .....	137
5.4.4	display 的问题区 .....	138
5.5	标签的合理嵌套 .....	139
5.5.1	标签嵌套基本规则 .....	139
5.5.2	错误嵌套时的表现情况 .....	139
5.6	SEO 搜索引擎优化——标签语义性 .....	144
5.6.1	为何要谈 SEO .....	144
5.6.2	SEO 是什么 .....	147
5.6.3	搜索爬虫工作原理 .....	147
5.6.4	爬虫抓取的是什么 .....	147
5.6.5	什么样的网站才能够被快速收录 .....	148
5.6.6	针对 SEO,前端开发要注意什么 .....	149
5.6.7	SEO 中表示强调的标签 .....	150
5.6.8	关于 SEO 的问题区 .....	150
5.7	嵌套层数与深度 .....	152
5.8	标签选择实战(1)——确定标签 .....	152
5.8.1	功能需求 .....	152





5.8.2	提出实现方案	153
5.8.3	标签选择思路分析	154
5.8.4	实现方案的对比分析	155
5.9	样式的可控性	155
5.9.1	原有选择器对样式的控制问题	155
5.9.2	加强版选择器——后代选择器	156
5.9.3	加强版选择器——子代选择器	157
5.9.4	加强版选择器——群组选择器	158
5.9.5	加强版选择器优先级算法	159
5.9.6	关于样式可控性的问题区	160
5.10	标签选择实战(2)——样式控制	162
5.10.1	方案1——使用ul无序列表	162
5.10.2	方案2——使用dl自定义列表	163
5.10.3	当前最优方案	164
第6章	模块布局(下)——可用性与扩展性	167
6.1	a 标签	167
6.1.1	超链接	167
6.1.2	超链接的属性	167
6.1.3	锚链接	168
6.1.4	超链接的基本样式	170
6.1.5	关于a 标签的问题区	171
6.2	光标样式	173
6.2.1	光标效果	173
6.2.2	cursor 相关属性	173
6.2.3	关于 cursor: hand	175
6.2.4	关于自定义光标样式的支持程度	175
6.3	标签选择实战(3)——添加链接	175
6.3.1	为实例添加a 标签	175
6.3.2	调整可触区	176
6.3.3	当前方案的具体代码	177
6.3.4	方案结束了吗	178
6.4	img 标签的选用	180
6.4.1	img 标签基本语法	180
6.4.2	数据图与背景图	181
6.4.3	img 问题的规避	181
6.4.4	img 中 alt 与 title 的区别	183
6.4.5	href 与 src 的区别	185
6.5	后台维护对前端的影响	185



6.5.1	图像加载对页面布局的影响	185
6.5.2	文字超出造成的页面混乱	186
6.6	网页中哪里需控制高度或超出隐藏	188
6.6.1	不同页面的不同需求	188
6.6.2	关于高度设定的基本结论	191
6.6.3	何时考虑超出隐藏	191
6.6.4	关于“高度控制与超出隐藏”的问题区	191
6.7	内容的超出处理——overflow	192
6.7.1	基本语法与功能	192
6.7.2	实现文本超出隐藏	192
6.7.3	实现文本超出显示为省略号	193
6.7.4	关于 overflow 的问题区	195
6.8	代码扩展性——关于 margin 负值	195
6.8.1	前后台数据整合方式	195
6.8.2	比数据条数少一个的虚线如何实现	197
6.8.3	特殊情况的类名设置详析	198
6.8.4	扩展性曾经的救世主——margin 负值	199
6.8.5	margin 负值的问题区	202
6.9	标签选择实战(4)——完成开发	204
6.9.1	考虑超出和 margin 负值	204
6.9.2	完整版代码	204
6.9.3	总结	206
第 7 章	文本等细节类样式处理	208
7.1	背景类样式	208
7.1.1	背景颜色——background-color 属性	208
7.1.2	背景图片——background-image 属性	209
7.1.3	背景重复——background-repeat 属性	210
7.1.4	背景定位——background-position 属性	211
7.1.5	背景关联——background-attachment 属性	211
7.1.6	复合写法——background 属性	213
7.1.7	背景类样式的相关问题	213
7.2	透明背景	214
7.2.1	opacity 与 filter	214
7.2.2	rgba 控制	216
7.2.3	transparent	217
7.2.4	透明背景的问题区	217
7.3	背景图合并	217
7.3.1	什么是背景图合并	217



7.3.2	为何进行背景图合并	217
7.3.3	背景图合并的核心技术与操作方法	218
7.3.4	CSS Sprite	219
7.3.5	背景图合并的问题区	220
7.4	段落样式	221
7.4.1	line-height	221
7.4.2	text-decoration	222
7.4.3	text-indent	222
7.4.4	text-align	224
7.4.5	vertical-align	225
7.4.6	word-spacing 与 letter-spacing	227
7.4.7	word-wrap 与 word-break	229
7.4.8	段落样式的问题区	231
7.5	字体类样式	231
7.5.1	color	231
7.5.2	font-family	232
7.5.3	font-size	233
7.5.4	font-style	233
7.5.5	font-weight	234
7.5.6	font 复合样式	234
7.5.7	网络字体	236
7.5.8	字体类样式的问题区	238
第8章	特殊布局情况——定位布局	240
8.1	定位属性	240
8.1.1	为何要使用定位	240
8.1.2	生活中的“定位”——便携贴	240
8.1.3	定位——position 属性	240
8.1.4	定位对文档流的影响	241
8.2	绝对定位的位置控制	244
8.2.1	设置绝对定位的元素的基本特点	244
8.2.2	定位——left 等属性	246
8.3	层级覆盖关系	249
8.3.1	定位——z-index 属性	249
8.3.2	多级别的层叠关系比较	251
8.4	固定定位	253
第9章	特殊布局情况——界限控制与伪元素的妙用	255
9.1	未设置固定宽高元素的界限控制	255



9.1.1	何处需要考虑界限控制	255
9.1.2	最小高度	255
9.1.3	最小宽度	257
9.1.4	最大宽度与最大高度	260
9.2	伪元素	260
9.2.1	什么是伪元素	260
9.2.2	伪元素的种类	260
9.2.3	伪元素的书写格式	261
9.2.4	after 与 before 伪元素	261
9.2.5	让伪元素按照块元素特性渲染	262
9.2.6	伪元素实现背景图	263
9.2.7	伪元素的问题区	263
第 10 章	表格	264
10.1	table 布局的兴衰	264
10.1.1	表格的发展历史	264
10.1.2	表格的应用	264
10.2	table 各类元素以及用法	265
10.2.1	基本标签	265
10.2.2	表格的嵌套规则	265
10.2.3	行列合并控制	265
10.2.4	关于表格元素的问题区	267
10.3	基本数据表的开发与制作	268
10.3.1	基本数据表的功能需求	268
10.3.2	基本数据表的实现思路	269
10.3.3	基本数据表的需求分析	269
10.3.4	基本数据表的实现	270
10.4	table 元素的属性	271
10.4.1	table 的常见属性	271
10.4.2	width 与 height——宽度与高度	272
10.4.3	border——表格边框设置	273
10.4.4	cellspacing 与 cellpadding——空白区设置	274
10.5	表格特有的 CSS 属性	275
10.5.1	合并单元格之间的边框——border-collapse	275
10.5.2	边框之间的空隙——border-spacing	276
10.5.3	空白单元格——empty-cells	276
10.6	表格属性与样式选用原则	277



第 11 章 表单 .....	278
11.1 认识表单 .....	278
11.1.1 表单的作用——实现对话 .....	278
11.1.2 向服务端发送数据的场景示例 .....	278
11.1.3 表单的基本结构 .....	279
11.1.4 各类表单元素通用属性 .....	280
11.2 表单常用元素 .....	280
11.2.1 form .....	280
11.2.2 input .....	282
11.2.3 label .....	286
11.2.4 select、option 与 optgroup .....	287
11.2.5 textarea .....	290
11.2.6 button .....	290
11.2.7 fieldset 与 legend 元素 .....	291
11.2.8 表单元素的问题区 .....	291
11.3 表单嵌套规则 .....	292
11.4 表单元素的特殊状态属性 .....	292
11.5 属性选择器 .....	294
11.5.1 属性选择器的应用场景 .....	294
11.5.2 基本的属性选择器 .....	294
11.5.3 模糊类属性选择器 .....	294
11.6 表单元素的实际应用 .....	296
11.6.1 去掉表单元素外部的聚焦线 .....	296
11.6.2 textarea 的尺寸控制 .....	297
11.6.3 自定义样式的表单元素 .....	297
第 12 章 停下来回头看路 .....	301
12.1 从○开始 .....	301
12.1.1 ○是什么 .....	301
12.1.2 最常见的答案 .....	301
12.1.3 让结果变得更优秀 .....	301
12.2 网状复习法 .....	303
12.2.1 网状复习法的特点 .....	303
12.2.2 网状复习法的实现方式 .....	303
12.2.3 网状复习法的简单案例 .....	304
12.2.4 网状复习 HTML 与 CSS .....	304
12.3 归纳整理法 .....	306
12.3.1 归纳整理法的特点 .....	306



12.3.2	归纳整理法的实现方式	306
12.3.3	归纳整理复习 HTML 与 CSS	306
12.4	hack 技术	310
12.4.1	什么是 hack 技术	310
12.4.2	常用 IE hack	311
12.4.3	IE 条件注释	311
12.5	实现网页开发之后要考虑的东西	312
12.6	PC 端浏览器的兼容问题	312
<b>第 13 章</b>	<b>HTML5 新标签与 CSS3 基础</b>	<b>313</b>
13.1	HTML5 新增元素	313
13.1.1	新增结构元素及含义	313
13.1.2	使用 HTML5 新结构元素完成页面搭建	314
13.1.3	HTML5 新元素的问题区	316
13.2	浏览器内核	317
13.2.1	浏览器主要内核	317
13.2.2	常见浏览器内核前缀	318
13.2.3	浏览器内核的问题区	319
13.3	CSS3 选择器	319
13.3.1	CSS2.0 选择器回顾	319
13.3.2	CSS3 选择器——通用兄弟选择器	319
13.3.3	CSS3 选择器——伪类选择器	320
13.3.4	CSS3 选择器的问题区	324
13.4	CSS3 圆角边框	325
13.4.1	圆角边框——border-radius	325
13.4.2	圆角边框效果实例	329
13.4.3	CSS3 圆角带来的变革	330
13.4.4	CSS3 圆角边框的问题区	333
13.5	CSS3 文本阴影	333
13.5.1	文本阴影——text-shadow	333
13.5.2	文本阴影效果实例	335
13.5.3	文本阴影的问题区	337
13.6	CSS3 盒阴影	337
13.6.1	盒阴影——box-shadow	337
13.6.2	盒阴影的效果实例	338
13.6.3	关于盒阴影的问题区	342
13.7	CSS3 背景新属性	342
13.7.1	背景尺寸——background-size	342
13.7.2	背景切割——background-clip	344



13.7.3	背景原点——background-origin .....	345
13.7.4	背景切割与背景原点的区别 .....	345
13.8	渐变背景 .....	347
13.8.1	什么是渐变 .....	347
13.8.2	渐变的种类 .....	347
13.8.3	如何书写 CSS3 渐变 .....	347
13.9	新元素和 CSS3 基础属性为网站带来了什么 .....	350
<b>第 14 章</b>	<b>转战移动端(上)——百分比与 rem .....</b>	<b>352</b>
14.1	移动端发展 .....	352
14.1.1	智能手机热潮 .....	352
14.1.2	针对移动端的探索 .....	352
14.1.3	分辨率初变 .....	354
14.1.4	多分辨率的处理 .....	354
14.1.5	移动端的未来 .....	355
14.1.6	移动端发展总结 & 开发移动端的基本流程 .....	355
14.2	设备调试方法 .....	356
14.2.1	设备调试方法的种类 .....	356
14.2.2	调试的基本原则：多台真机测试 .....	357
14.3	视口——viewport .....	357
14.3.1	视口以及常见数值 .....	357
14.3.2	调整视口大小——命令类 .....	357
14.3.3	viewport 元标签以及属性 .....	358
14.3.4	视口调整的各种命令 .....	358
14.3.5	对待视口的基本原则 .....	359
14.4	当盒模型与行高遇到百分比 .....	359
14.4.1	盒模型单位如何选择 .....	359
14.4.2	margin 和 padding 使用百分比作为单位 .....	359
14.4.3	height 使用百分比作为单位 .....	363
14.4.4	border 使用百分比作为单位 .....	364
14.4.5	line-height 使用百分比作为单位 .....	365
14.5	CSS3 新增度量单位 .....	366
14.5.1	新度量单位 .....	366
14.5.2	rem 与 em .....	366
14.5.3	vw 与 vh .....	367
14.6	字体处理不容小觑 .....	367
14.6.1	美工图设计的基准字体 .....	367
14.6.2	移动端网络字体使用更加频繁 .....	368
14.7	盒阴影的妙用 .....	368



14.8	背景图的处理	368
14.9	使用 JS 配合 rem 让页面适应各个分辨率	368
14.9.1	Step1 查看设计图,确定需要兼容的分辨率	368
14.9.2	Step2 调整视口	369
14.9.3	Step3 确定设计图的最小字体	369
14.9.4	Step4 按照设计图的像素进行开发	369
14.9.5	Step5 使用百分比和 rem 替换 px	371
14.9.6	Step6 使用 JS 控制基准字体	373
<b>第 15 章</b>	<b>转战移动端(下)——响应式 &amp; 移动端的探索</b>	<b>376</b>
15.1	响应式布局	376
15.1.1	理解响应式布局	376
15.1.2	响应式布局的优劣势	376
15.1.3	响应式布局的核心技术	377
15.2	媒体查询	377
15.2.1	什么是媒体查询	377
15.2.2	媒体查询书写方法	377
15.2.3	常见媒体类型	378
15.2.4	关于媒体的特性	378
15.3	让移动端开发变得更好——关于高清屏幕	379
15.3.1	高清分辨率	379
15.3.2	此前移动端开发存在的一些不足	379
15.3.3	按照高清分辨率解读的“想法”	379
15.3.4	flexible.js 的用法	380
15.3.5	flexible 框架使用的注意事项	381
15.4	让移动端开发变得更快——固定像素的实现方法	381
15.4.1	MetaHandler.js	381
15.4.2	框架当前还存在的问题	382
15.5	移动端兼容	383
15.5.1	CSS3 媒体查询兼容问题	383
15.5.2	HTML 与 CSS 的基本兼容问题	384
15.5.3	默认样式处理	385
<b>第 16 章</b>	<b>CSS3 变形与动画</b>	<b>387</b>
16.1	CSS3 二维变形	387
16.1.1	二维变形基本语法	387
16.1.2	具体变形方式语法详析	387
16.1.3	变形顺序对最终结果是否会造成影响	390
16.2	CSS3 三维变形	391





16.2.1	如何触发三维变形	391
16.2.2	Z 轴的位置	392
16.2.3	三维变形的变形属性	393
16.2.4	视角	394
16.2.5	旋转带来的问题	394
16.2.6	关于三维变形的应用	394
16.2.7	关于变形的问题区	395
16.3	CSS3 过渡	395
16.3.1	过渡的基本属性	395
16.3.2	过渡的合写方法 transition	396
16.3.3	多属性过渡时,各个属性的书写方法	396
16.3.4	过渡得以实现的必备要素	397
16.3.5	关于过渡的问题区	398
16.4	CSS3 动画	399
16.4.1	帧与关键帧	399
16.4.2	CSS3 动画的基本语法	399
16.4.3	animation 动画的基本属性	400
16.4.4	动画命令的合写方法 animation	401
16.4.5	动画与过渡的比较	401
第 17 章	各章节自评习题集	402
17.1	习题集 01	403
17.1.1	习题内容	403
17.1.2	习题答案	404
17.2	习题集 02	404
17.2.1	习题内容	404
17.2.2	习题答案	406
17.3	习题集 03	407
17.3.1	习题内容	407
17.3.2	习题答案	410
17.4	习题集 04	411
17.4.1	习题内容	411
17.4.2	习题答案	412
17.5	习题集 05	417
17.5.1	习题内容	417
17.5.2	习题答案	418
17.6	习题集 06	419
17.6.1	习题内容	420
17.6.2	习题答案	421





17.7	习题集 07 .....	422
17.7.1	习题内容 .....	422
17.7.2	习题答案 .....	424
17.8	习题集 08 .....	425
17.8.1	习题内容 .....	426
17.8.2	习题答案 .....	428
17.9	习题集 09 .....	431
17.9.1	习题内容 .....	431
17.9.2	习题答案 .....	432
17.10	习题集 10 .....	432
17.10.1	习题内容 .....	432
17.10.2	习题答案 .....	433
17.11	习题集 11 .....	434
17.11.1	习题内容 .....	434
17.11.2	习题答案 .....	435
17.12	习题集 12 .....	436
17.12.1	习题内容 .....	436
17.12.2	习题答案 .....	437
17.13	习题集 13 .....	438
17.13.1	习题内容 .....	438
17.13.2	习题答案 .....	439
17.14	习题集 14 .....	439
17.14.1	习题内容 .....	439
17.14.2	习题答案 .....	441
<b>第 18 章</b>	<b>各章节代码实战 .....</b>	<b>444</b>
18.1	代码检错——[第 2、3 章] .....	444
18.1.1	实战题目 .....	444
18.1.2	实战答案 .....	445
18.2	为元素设置盒模型样式——[第 3 章] .....	446
18.2.1	实战题目 .....	446
18.2.2	实战答案 .....	446
18.3	使用浮动实现网页布局——[第 4 章] .....	447
18.3.1	实战题目 .....	447
18.3.2	实战答案 .....	449
18.4	合理选择标签——[第 5、6 章] .....	455
18.4.1	实战题目 .....	455
18.4.2	实战答案 .....	456
18.5	文本样式处理——[第 7 章] .....	458





18.5.1	实战题目	458
18.5.2	实战答案	459
18.6	定位布局——[第 8 章]	460
18.6.1	实战题目	460
18.6.2	实战答案	461
18.7	伪元素的应用——[第 9 章]	462
18.7.1	实战题目	462
18.7.2	实战答案	463
18.8	课程表与表格简历制作——[第 10 章]	465
18.8.1	实战题目	465
18.8.2	实战答案	466
18.9	CSS3 阴影特效——[第 13 章]	471
18.9.1	实战题目	471
18.9.2	实战答案	472
18.10	响应式布局——[第 15 章]	474
18.10.1	实战题目	474
18.10.2	实战答案	476
18.11	二维三维特效动画——[第 16 章]	477
18.11.1	实战题目	477
18.11.2	实战答案	479
第 19 章	网页案例实战	491
19.1	PC 端网页开发实战	491
19.1.1	功能需求	491
19.1.2	整体测绘	491
19.1.3	实现整体布局	494
19.1.4	实现头部模块(含 LOGO 与微信)部分	496
19.1.5	实现导航模块部分	497
19.1.6	实现内容左侧最新文章部分	498
19.1.7	实现内容右侧广告区部分	502
19.1.8	实现 tab 区域标题部分	504
19.1.9	实现 tab 区域内容部分	505
19.1.10	实现底部(版权)区域部分	507
19.1.11	实现大图部分	508
19.1.12	考虑超出隐藏以及光标移入状态	510
19.1.13	考虑临界值	513
19.1.14	代码优化	513
19.2	移动端网页开发实战	514
19.2.1	功能需求	514





19.2.2	整体测绘 .....	514
19.2.3	实现整体布局 .....	517
19.2.4	添加 JS 控制,实现多分辨率自适应 .....	518
19.2.5	实现热门与推荐部分 .....	519
19.2.6	实现具体文章模块部分 .....	522
19.2.7	实现顶部区域 .....	524
19.2.8	实现底部版权部分 .....	527
19.2.9	代码优化 .....	527
<b>第 20 章</b>	<b>附录 .....</b>	<b>528</b>
20.1	HTML5 发展史 .....	528
20.1.1	萌芽 .....	528
20.1.2	第一次浏览器大战 .....	528
20.1.3	第二次浏览器大战 .....	528
20.1.4	第三次浏览器大战 .....	529
20.2	手机端操作系统发展史 .....	529
20.2.1	诺基亚的世界开始动摇 .....	529
20.2.2	微软、苹果、谷歌之战 .....	530
20.2.3	一代霸主陨落 .....	530
20.2.4	iOS、安卓高奏凯歌 .....	530
20.2.5	苹果和 Adobe .....	530
20.3	HTML 的各种布局 .....	531
20.3.1	表格布局 .....	531
20.3.2	DIV+CSS .....	531
20.3.3	960 栅格 .....	531
20.3.4	经典的三栏布局——双飞翼 .....	531
20.3.5	瀑布流布局 .....	532
20.3.6	响应式布局 .....	532
20.3.7	单页面无限滚动 .....	533
20.3.8	移动端的 rem 自适应布局 .....	533
20.3.9	其他布局以及未来 .....	533
20.4	编辑器插件安装与应用 .....	533
20.4.1	插件安装 .....	533
20.4.2	关于 Emmet 插件的相关配置 .....	534
20.4.3	利用 Emmet 插件快速编写 HTML 代码 .....	535
20.5	开发需要准备的基本软件 .....	537
20.5.1	基本软件列表 .....	537
20.5.2	WAMP 软件的安装 .....	537
20.5.3	初识多人协同开发用的“版本控制管理工具” .....	541



20.6	HTML5 学习平台与网站 .....	544
20.6.1	HTML5 布局之路——官方平台 .....	544
20.6.2	其他学习类网站 .....	544
20.6.3	找工作的网站平台 .....	544
20.7	单词列表 .....	545
20.8	网页中部分模块的 CSS 命名参考 .....	545
20.9	重置代码解析 .....	546
20.9.1	重置代码 .....	546
20.9.2	重置代码解析 .....	547
20.9.3	在重置文件中添加的语句 .....	548
20.10	开发备忘录 .....	548
20.10.1	书写基本的需求分析报告 .....	548
20.10.2	基本的前期准备工作 .....	549
20.10.3	移动端与 PC 端的特殊性 .....	549
20.10.4	整体开发的基本顺序提醒 .....	549
20.10.5	具体开发中的注意事项 .....	550
20.10.6	其他 .....	550
20.11	Iconfont .....	551
20.11.1	什么是 Iconfont .....	551
20.11.2	Iconfont 中图标下载 .....	551
20.11.3	Iconfont 可能出现的问题以及解决办法 .....	552
20.11.4	实际开发中 Iconfont 的用途 .....	552
	参考文献 .....	553





# 第1章 旅途之前



## 1.1 讲解方法与旅途指南

### 1.1.1 和其他技术书籍相比,本书到底有何不同

#### 1. 非字典式的排布方式

本书并没有按照“自上而下罗列标签”这种“字典”的模式进行讲解,而是依照“实战流程”的顺序安排章节和知识。将开发过程拆分出来,按照开发中的一步步操作,讲解每一步相关知识和技术。伴随着学习者的阅读、学习和操作,一步步实现符合前端开发规范的网站的开发与制作。

#### 2. 选择性的知识呈现

随着时代的发展,在 HTML5 技术的知识体系当中,有不少知识都已经被淘汰;在实际开发当中,也有不少知识并不被广大开发者所使用。学习技术是为了真正地应用于实战当中,而非单纯的理论记忆,因此,基于 HTML5 开发工程师工作的内容,本书对知识进行了整理和筛选,选择性地呈现于本书当中。

#### 3. 技术也能通俗易懂

本书的很多地方借助了生活中的实例,让技术变得通俗易懂,使距离我们很远的技术就如同在身边一样。此外,针对部分知识设置了问题区,归纳总结了实战开发当中容易出现的一些问题,并给出了相应解答。

#### 4. 知识讲解、基础习题、代码练习、案例实战四合一

学习知识之后,需要巩固理论知识,进行代码练习与实战,最后则应尝试开发实际的网页。

本书针对大部分章节的技术知识,提供了相对应的理论练习题和代码练习题,还提供了涉及多章节知识的“网页案例实战”,让读者能够从理论出发,逐步完成知识的认识、记忆、理解和应用。

### 1.1.2 书籍概览

本书可分为 6 个部分,如表 1.1 所示,第一部分主要讲解 PC 端(桌端)的网页开发;第

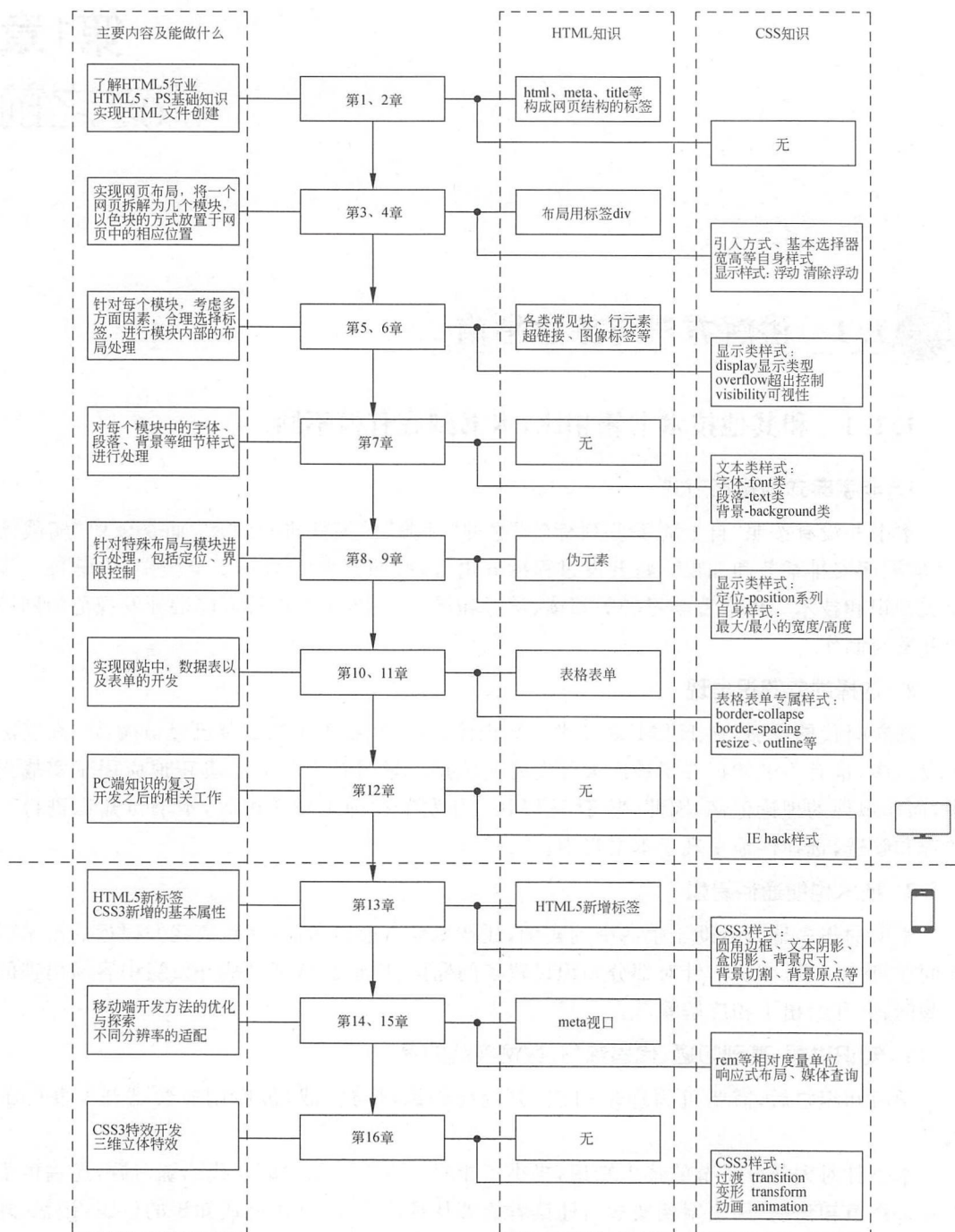


图 1.1 旅途指南





二部分主要讲解移动端的网页开发；从第三部分到第五部分，分别是与前两部分配套的理论知识练习题、实战知识练习题、网页开发实例；最后一部分为附录，包括行业知识、开发工具、开发备忘录等内容。

表 1.1 书籍模块划分以及内容概要

模块划分	内 容 概 要	涉 及 章 节
第一部分	PC 端开发知识与技术	第 1~12 章
第二部分	移动端开发知识与技术	第 13~16 章
第三部分	基础知识自评习题集	第 17 章(与前 16 章部分章节相对应)
第四部分	代码练习题	第 18 章(与前 16 章部分章节相对应)
第五部分	涉及多章节知识的网页案例实战	第 19 章(与前 16 章部分章节相对应)
第六部分	各类软件、工具、辅助学习类文件	附录

### 1.1.3 推荐学习流程

对于初学者，建议按照如下流程进行知识的学习：

某章节理论知识→章节对应的基础知识自评习题(第 17 章)→章节对应的代码实战练习(第 18 章)→在一部分章节学习之后，进行较大型的网页案例实战(第 19 章)。

对 HTML5 有所了解的学习者，可以根据自己开发过程中的薄弱点，选择章节进行针对性学习。

### 1.1.4 旅途指南

本书的第一、二部分(前 16 章)是技术知识章节，在图 1.1 中，包括前 16 章的主要知识梗概，相应章节包含的 HTML 知识和 CSS 知识，以及学习相应章节后能够做什么。



## 1.2 HTML5 学习的前期准备

在开启 HTML5 布局之路之前，有一些知识是需要掌握的，良好的基础知识能够使日后专业知识的学习变得轻松，也能够规避实际开发中“影响开发速度”的一些问题。

学习的前期准备工作主要包括：

- (1) 了解什么是 HTML5，不要盲目地进行学习；
- (2) 了解行业的发展规律，将目光放长远，并且充分理解供求关系对行业的影响；
- (3) 掌握 HTML 与 CSS 的学法，便于提升自己的学习效率；
- (4) 掌握 Photoshop 的使用，实现页面的快速切图；
- (5) 打字速度越快越好，建议达到 100 英文字母/分钟；
- (6) 掌握使用 Sublime Text 编辑器进行代码的编写，提升代码的编写速度。



## 1.3 认识 HTML5

### 1.3.1 HTML5 是做什么的

HTML5 用于实现人们日常看到的所有网站(网站中的模块布局、文字段落图片的样式、动作等),但是不涉及后台和数据层面(具体的图片、文字是什么)。

换言之,HTML5 开发工程师是负责将设计师设计好的网页图片(通常是 PSD 格式),用代码实现出来,包括在网页的某个位置放置一个块,给一个块设置颜色,调整字体大小,让图片动起来等操作。

### 1.3.2 HTML5 的由来

不熟悉 HTML5 的人,可能会很熟悉 2005 年以前常用的一个专业名词——网页设计与制作。在 2005 年之前,开发工程师与设计师的工作是二合一的,直至 2005 年 JavaScript 的崛起,行业发生了比较大的变化,社会化分工越来越明细,网站的制作越来越受到人们的重视,设计和制作自此分开,“Web 前端开发工程师”这个词语开始出现在各个一线城市。2008 年,HTML5 横空出世,2009 年,HTML5 这个全新的词语在北京的一些顶尖级公司出现,2012 年逐渐普及开来,2014 年得以迅速发展。

### 1.3.3 HTML5 的基本组成

#### 1. HTML5 = HTML + CSS + JavaScript

HTML5 并不仅仅指代一种语言,它是三种语言的结合体,分别是 HTML、CSS 和 JavaScript。

其中,HTML 是一种脚本语言,用于构建网页当中的结构;CSS 称为层叠样式表,用于处理网页中每个标签的样式;JavaScript 也简称为 JS,用于处理网页中标签的行为。

#### 2. 关于结构、样式、行为的理解

结构——在整个网页中有标题、列表、图片等。

样式——标题文字的字体大小、颜色、字体;图片的大小;某个块的背景色或背景图等。

行为——在网页上四处飘动的广告;图片滚动;浏览某购物网站当中的商品,光标移动到商品上时,商品被放大的效果等。

如果将网页比作一个房间,后台的数据就相当于是具体的物品,HTML 是包含各个物品的容器,CSS 用于控制容器的位置,还能够针对物品进行各种装饰和美化,JS 则是为各个容器添加一些功能和动作。

### 1.3.4 专业化角度认识 HTML5

比较通俗地理解了 HTML5 之后,下面从比较专业的角度来谈谈 HTML5。





### 1. 狭义角度(从版本层面而言)

HTML5 是 W3C 制定的关于 HTML 技术约定的新规范,可以简单地认为 HTML5 是原有 HTML4 的升级版,升级的内容涵盖了 HTML5 新属性、CSS3 技术、JavaScript 方面的一些 API 等。

### 2. 广义角度(从行业层面而言)

随着 HTML5 的发展,HTML5 技术逐渐从单纯的技术形成了一个行业,而 HTML5/H5 也成为这个行业的代名词,也可以作为一种行业开发的技术标准。此外,HTML5、VR、AR、大数据等都是当前各类新技术新行业的代名词。

### 3. 技术角度(从单纯的技术变化而言)

从技术层面来说,HTML5 相比之前的 HTML4 技术,主要有以下 5 个方面的变化。

- (1) 新增了 HTML5 的相关标签及属性;
- (2) 在视觉层面,添加了 CSS3 新特性,让特效从单纯的二维平面发展到三维立体;
- (3) 增加了移动端的开发与制作;
- (4) 增加 Canvas,使绘图功能更加强大,能够使用其进行一些图形的绘制和页面游戏的开发;
- (5) 在 JavaScript 方面,增加了大量与设备交互的 API(如地理定位等),还将技术扩展到更深层的方面(Web Workers、Web Socket 等)。

**备注:** 本书会详细介绍这些技术变化中的前三点;由于 Canvas 以及 JavaScript 方面的技术与布局无关,其更多涉及的是逻辑层面,在本书当中并没有提及,如果感兴趣可以在掌握 HTML 与 CSS 基本知识之后,学习 JavaScript 技术。

## 1.3.5 容易混淆的概念

### (1) 移动端的 HTML5 == 应用程序开发语言?

对于 iOS 系统的苹果手机,其中的应用程序采用的开发语言为 Object C;对于安卓系统的手机,其中的应用程序采用的开发语言为 Java。HTML5 主要开发的是移动端的网页、微信平台中的移动端网页。HTML5 行业中说的 Web APP,指的是采用 HTML5 技术,进行“仿应用程序体验”的网站开发。

移动端的 HTML5 并非应用程序的开发语言,不过,HTML5 的确可以与其他语言(Java、Object C 等)混合开发 APP,我们把这种使用了多种技术语言开发的 APP 称为“Hybird APP”,而把单纯使用原生语言开发的 APP 称为“Native APP”。

### (2) 静态页面 == 没有 JS 的前端页面?

有些人根据一个网页中的元素是否运动来区分这个网页是静态页面还是动态页面,认为网页中使用 JS,让元素能够运动的是动态页面,否则是静态页面。这种理解是错误的。所谓静态页面,指的是没有动态数据的页面,也就是纯前端页面,无论有没有 JS,都将这种前端页面称为静态页面;所谓动态页面,指的是前后台整合之后的页面(后缀名通常为 .php、.java 等),网页中的数据并非前端书写,而是从后台数据库传递过来的。

### (3) HTML5 开发 == 网页设计与制作?

虽然 HTML5 开发是由原来的网页设计与制作演变而来的,但是这两者有着很大的区

别,并不能够随意画等号。

网页设计与制作,主要针对 PC 平台,会涉及一部分设计工作,还需要完成从设计图到网页的实现,使用的开发工具是网页三剑客——Photoshop、Flash、Dreamweaver。

随着时代的发展,行业的发展使得“网页设计与制作”这一职业逐渐遭到了淘汰,原因主要有以下 4 个。

① 网页设计与网页制作是两个完全不同的领域,前者由美感主导,后者则由逻辑思维主导。对于开发人员来说,如果将宝贵的精力分散到两个不同的行业中,最后通常两方面都是半斤八两,没有实质的竞争力。

② “网页设计与制作”中的制作,指的是网页的结构与样式(即 HTML+CSS)以及少量的 JavaScript。而行业向前发展之后,网站中 JavaScript 已经占据了极大的比重,如果还停留在原有的结构和样式中,发展空间会变得很小。

③ 网页设计与制作当中的结构实现,通常采用的是 table 布局;而 Web 前端开发工程师、HTML5 结构的实现,采用的是 HTML+CSS(也有称为 DIV+CSS)方式的布局,因此, Dreamweaver 工具的使用也就没有什么必要性了,取而代之的是内存占用小、开发速度快的文本类编辑器。而 Flash,在与 HTML5 的大战当中战败,当前已经退出了移动端以及电视平台的市场争夺,在 PC 平台的应用也越来越少。换言之,Flash 在网页制作的领域里已经江郎才尽。原来的网页三剑客只剩下一个 Photoshop,对于图像的处理,在前端工程师的工作要求中,“掌握切图功能”即可。

④ 移动互联网的飞速发展,也使得 HTML5 的地位迅速提升。移动设备有其特殊的开发要求,原有的网页设计与制作,早已无法满足开发的需要。



## 1.4 了解 HTML5 行业前景与基本概念

### 1.4.1 一些行业词汇

#### 1. 页面重构

页面重构,是前端开发再细化的一种工作,与之相对应的职位为“页面重构师”,多存在于工作分类比较细化的大型公司。页面重构师的主要工作是把 PSD 源文件(美工图)转换成基本的网页(切图工作),进行 HTML 与 CSS 代码的编写,较少涉及 JavaScript。单纯的页面重构,所涉及的工作内容一般是“分析设计稿→切图→书写 HTML 和 CSS”。

#### 2. W3C

W3C 的全称为“World Wide Web Consortium”,中文含义为“万维网联盟”,是 Web 技术领域具有一定权威和影响力的国际中立性的技术标准机构,主要进行标准的制作与开发,例如,制作 CSS 标准规范、HTML5 标准规范等。

### 1.4.2 HTML5 的行业并非一劳永逸

#### 1. 当前行业好坏并不能够决定未来

入行之后,一个开发工程师在向更高职位、薪资迈进时,会经过公司层层筛选,在发展



这条路上能够真正走得好的并不多。即便是能够在行业很好的时候进入这个行业,也并不等同于而后的几十年会发展得好。也有在最初行业不是太景气的时候进入行业,但最终依旧在这个行业杀出一条“血路”来。

敲门砖的好坏决定着起点的高低,但是决定未来的是当下和未来的自己。经典的二八法则是很适用于这里的,只有 20% 的人能够在工作中走到更高的职位或拿到更可观的薪资。想要在未来突出重围,那么就必须努力让自己成为那 20%。

## 2. 最核心的能力与最重要的思想

所有能力当中学习能力是最重要的,没有之一。经验没有,可以学。不会团队协作,可以学。学习能力强,其他方面能力的欠缺都是暂时的(人品除外)。

最重要的思想是主动思考,思考如何做得更好,思考除了技术之外自己还能够收获什么,思考目标,思考意义,思考工作中自己的心态与不足。

还没有进入行业的学习者,试问这些问题是否想明白了:技术之外我还要掌握什么?我的目标是什么?我希望我的未来是什么样子?在这个行业当中我要做到什么程度?为何在学的时候我的速度比较快/慢?我的学习方法有什么问题?行业发生变化时,我应该如何调整?(在这里仅罗列一部分问题,可以根据这些问题的反思发现自己有没有主动去“思考”。)

### 1.4.3 近几年来 HTML5 行业的变化

2012 年的时候,移动端刚刚起步,那时项目开发使用的是 px 这种绝对度量单位。而随着行业的发展,逐渐出现了响应式布局(2013 年年初流行起来),百分比与 em(2013 年下半年)、rem(2014 年)的开发方法开始盛行;2015—2016 年,各类框架和实现方法层出不穷。

换言之,随着时间的推移,以前的“新”事物已经变成了“旧”东西,而行业对这个事物的认识和了解程度越来越强,相对来说对技术开发人员的整体要求越来越高。

### 1.4.4 到底是什么决定着薪资

决定薪资的标准是老板是否是土豪?还是自己能力技术有多强?还是应聘者的学历?

都不是!很多人觉得学历高的人通常综合能力更强,理应得到更高的工资。很多人觉得自己能力技术比别人强一些,所以能够拿到相应较高的薪资。但是真的是这样子吗?

从经济学的角度来说:成本无法决定价格!

在面试同样的岗位时,你在学业上投资了比别人更多的时间和金钱,是否应该拿更高的薪资?就如同你花三年学会了 HTML5 开发;别人花了 9 个月学会了 HTML5 开发,假设你们当前的技术水平是一样的,那么此时,公司在雇用你的时候要开出比别人高三倍的薪水吗?

再来说说技术能力。2012 年,北京 HTML5 开发工程师的平均薪资只有 4~4.5 千元,只有本科、计算机类专业的学生才有可能突破 5 千元大关。而那时,稍微掌握 iOS 的工程师,在北京都能够要到 6~8 千元的薪资。三年过去,到了 2015 年上半年,培训学习 iOS,技术水平很不错的学生,也仅仅能够拿到 6 千元左右的薪资,而 HTML5 则飙升到了 8~10 千元,一发不可收拾。到了 2016 年,虽然 HTML5 初学者的整体技术水平在上涨,但是薪资反而开始下降。

请记住：成本不决定价格，市场供需才决定价格。当职位多而人少的时候，每个开发人员都是香饽饽；当行业需求降下来了，而懂得该技术的人数多了的时候，就如同千军万马过独木桥……

#### 1.4.5 是谁决定你的去留

即便你已经入行开发工程师，也不能掉以轻心，因为你必须清楚是谁决定你的去留。

老板？自己？都不是！

所有的公司都是要盈利的，当一个工程师和别人拿着同样的薪资，创造的价值却不及别人的十分之一时，那么，距离被辞退就不远了，因为如果不把你请离，就是对其他同事的不负责。

公司衡量一个人的价值标准是什么呢？没错，就是和你拿着同样工资的这个“别人”！换言之，真正决定一个工程师去留的，是同岗位的其他竞争者。

每个企业都对某个岗位的人员存在一个预期，这个预期是由“其他竞争者平均水平”所决定的标准，这种标准并不是大家的随性而为，而是伴随着雇主和雇员长期市场博弈逐渐形成的。

如果雇员的能力低于这个预期，那么上级通常要么请他离开，要么让其换岗尝试其他工作（而其他工作通常都是比当前工作更基础，标准更低的）。

对于已经在行业中工作了一段时间的工程师，如果自身能力水平还没有得到足够成长的情况下，随着新人的不断涌入，其更强的技术与更低的工资价格以及更高的工作效率使得“老员工被请离”变成了时间问题。



### 1.5 HTML 与 CSS 的学法

#### 1.5.1 方法 1 整体到局部，骨架到血肉

在学习 HTML 和 CSS 时，会涉及网页的搭建。本书的讲解方式，也采用了这个方法，“由外及内”，“由整体到部分”，“由全局到细节”。

学习东西，特别是在初识某个事物时，一定要从主干到枝叶，而不要陷入细节、纠结于其中。主干如同知识的一个主线，这种先主干后枝叶的学习方法能够大大提升学习效率，也会更容易建立知识与知识间的关系，从而降低知识的遗漏量。

#### 1.5.2 方法 2 类比

本书还采用了另一种知识描述的方法：辨析，或者也可以叫作类比。这种方法主要针对相似的两种或多种事物，如 strong 与 em、块元素与行元素。

对于此类知识，建议多多思考，敲代码查看效果并根据实际情况进行总结，抓取几种事物的不同点，分成类别进行记忆。

#### 1.5.3 方法 3 记忆很重要

很多人在学习过程中觉得理解最重要，不需要记忆；也有人会觉得敲敲代码就熟悉了，



也不需要记忆。

必须承认,理解一个知识点的确很重要,它能够让我们更好更快地应用知识和解决代码问题,不过如果知识记不清楚,即便学习时理解了,应用时忘记了又有什么用呢?多敲代码虽然能够提升知识的熟练度,但是也存在不足的地方。敲代码实现一些网页时,大部分的页面是类似的,使用到的知识也比较局限,很容易造成熟悉开发常用知识,而不熟悉在开发中使用较少的知识。

因此,在学习时如果遇到大量的知识散点(如有哪些标签元素),建议还是要将其记忆清楚,之后再进行理解和应用。

#### 1.5.4 方法4 聚沙成塔

在网站中能够看到各种各样的效果,有些效果看上去很高大上,很炫美。然而,再炫美的效果也是由众多知识点组合而成的。当对效果抽丝剥茧,就会发现,其实最初的它很简单。

在生活当中,一朵樱花并不起眼,但是当数百棵樱花树的花瓣纷纷扬扬飘洒下来时,粉色的花雨无比美丽。

这就是所谓的聚沙成塔,在学习时,不要小瞧零碎的小知识点,因为正是由它们组成了千万网页。当我们希望制作出炫美的效果,或遇到一个看上去“很复杂”的网页时,也请记得这个道理,不要着急,也不要郁闷,针对“大效果”、“大场面”的页面进行分析、拆解,然后一步一步地去实现。

#### 1.5.5 方法5 循序渐进

一口吃不成胖子,也不可能仅仅不吃一顿饭就减肥成功。在学习过程中,会遇到一些“大型”的知识点,这种知识点比较难消理解。遇到此类知识点,不要想着如何一口吃掉它,而要循序渐进。

在循序渐进中,思路很重要,换句话说,我们知道一个知识点很“大”,也知道要一口一口地吃,一点儿一点儿地消化,但是,如果弄不清楚先吃什么再吃什么,也很难把这个知识点消化。这时候需要“思路”,也就是“流程”,在学习知识中,重点是关注流程和思路,而具体的小知识点充当的是血肉(这里和前面提到的第一个方法就挂钩了,就是分清主干和枝叶)。

#### 1.5.6 方法6 知识的迁移

用已有知识辅助未知知识的理解,是很好的一种学习方法,这也是建立知识间相互联系的一种很好的方法,通过这种“关系”的构建,能够让自己的知识变得更牢固。通常这种方法应用于拥有相似特点的事物。例如,圆角边框与外边距、背景切割与背景原点等(第13章中的知识)。

#### 1.5.7 方法7 生活辅助学习

利用生活中实际的事物去辅助抽象知识的学习,能够让我们更好更快地理解和吸收知识。这种方法在本书也有大量用例,例如,在讲解盒模型时借用了快递中的鱼缸,在讲解定位时借用了便携贴。

在理解一个抽象概念时,可以尝试在生活中找一种合适的例子,辅助自己理解,慢慢地,你将形成自己的学习方法。如果能够建立不同类型事物之间的联系,知识就变得更不容易遗忘了。当然,在使用这个方法的时候要注意:选用合理贴切的例子,从而防止被错误案例误导。

### 1.5.8 方法8 实践出真知

在学习一些代码时,有些人不喜欢动手,而更多的是喜欢看、听或者背,在此本书并不推荐这种思路。

对于代码类的知识,是个需要动手的活儿,掌握代码靠的不仅是记忆,还需要实践。尝试书写代码,发现现象或问题,然后归纳总结,形成理论并记忆。

换句话说,理论来源于实践,高于实践(高于实践的原因在于有总结、归纳、自我的加工)。在学习过程中,不仅要能够“将理论应用于实践”,也要“通过实践得出理论”。



## 1.6 开发工程师与 Photoshop

### 1.6.1 图片切图

#### 1. HTML5 开发工程师使用 Photoshop 做什么

前端与 Photoshop 的关系很密切,设计师使用 Photoshop 把网页的图片效果制作出来,而开发工程师要做的是:寻找出 PSD 图中的图片元素,将其切下来(简称切图)之后进行适当的处理,将图片存储为合适的格式,有时还需要进行背景图合并的操作。

特别注意的一点是:网页设计并不是开发工程师做的工作。

#### 2. 切图与制作的基本理念

开发工程师不能够擅自修改网页设计图。当出现和设计图不同的意见或观点时,开发工程师应当及时与设计人员沟通,可以向设计人员提出建议或意见,但最终决定要根据产品需求以及设计人员的意见而定。

对于确定后的设计图,开发工程师要做的是百分百地还原设计图,不能说“差不多”,一个“高档漂亮”的网页往往就是因为开发工程师的“差不多”变成了一个“糟乱差”的网页。

### 1.6.2 认识软件

在此以 Photoshop CS6 为例,讲解一下 Photoshop 的基本用法。如图 1.2 所示为 Photoshop 界面。

- (1) 最顶部为菜单栏,Photoshop 的所有操作在菜单栏当中都能够找到。
- (2) 界面左侧为工具栏,里面放置了各类 Photoshop 中常用的工具。
- (3) 在工具栏当中选择某一种工具之后,在③的位置,会出现关于这种工具的“工具菜单”。
- (4) 右侧部分是多个面板,每个面板有各自的功能,单击缩略图能够打开这个面板,安装 Photoshop 之后,有些面板可能是隐藏的,此时可以通过菜单当中的“窗口”选项,将需要



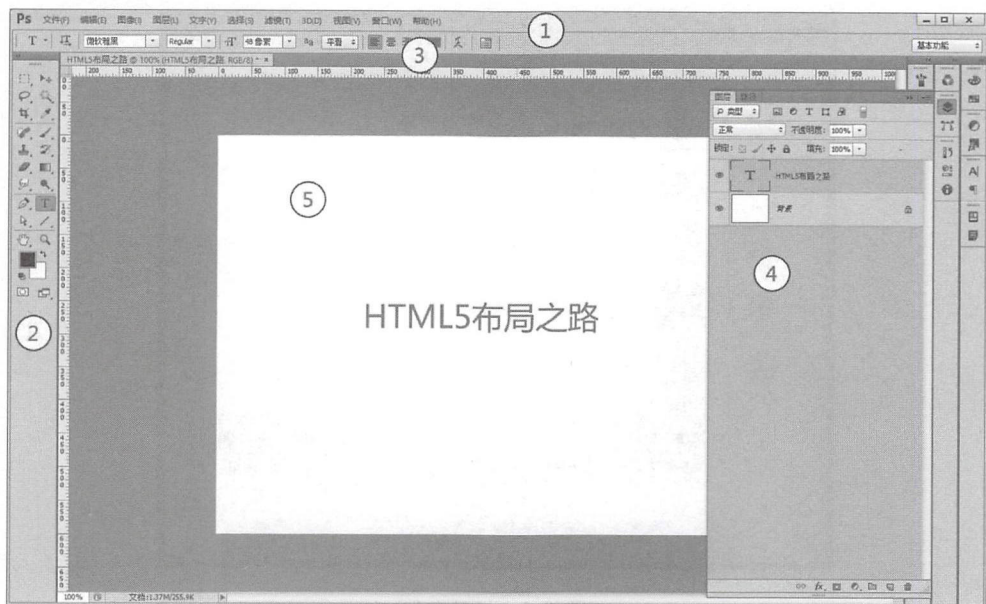


图 1.2 Photoshop 界面

的面板“调出来”。

(5) 中间的部分为内容区,是画布所处的区域。



## 1.7 切图与 Photoshop 相关用法

### 1.7.1 切图的基本流程

- (1) 打开已有 PSD 文件并看懂 PSD 图;
- (2) 找到切图目标;
- (3) 进行目标图层的相关操作(整理好要处理的图层);
- (4) 新建 PSD 文件用于存储目标图像;
- (5) 在新文件中调整图像位置;
- (6) 图片存储(需存储为合适格式);
- (7) 图片大小的处理与压缩。

### 1.7.2 打开文件

打开文件的方法有多种,可以将图像文件拖曳到 Photoshop 当中,也可以使用菜单栏中的“文件”→“打开”,也可以使用快捷键 Ctrl+O。

能够使用 Photoshop 打开的文件有两类,一类是 PSD 图,一类是平时看到的 JPG、PNG 图。

选择菜单栏中的“文件”→“打开”,然后寻找到要使用 Photoshop 打开的文件,单击“打开”按钮,之后会弹出“打开”对话框,如图 1.3 所示。



图 1.3 打开文件

### 1.7.3 找到切图目标

#### 1. 看懂 PSD 图

##### 1) 简单地理解图层

设计师在设计一个网页时,会将网页中的各个素材、模块,放置在不同图层当中,通过图层的叠加,形成完整的图像。

设计师的操作就如同在一张张透明的玻璃纸上作画,透过上面的玻璃纸可以看见下面纸上的内容,但是在任何玻璃纸上如何涂画都不会影响到其他玻璃纸,上面的玻璃纸会遮挡住下面的图像。最后将多张玻璃纸叠加起来,通过移动各个玻璃纸的位置或者添加更多的玻璃纸可改变最后的合成效果(即为 PSD 图),而构成这个效果的每一张玻璃纸就是一个图层。

##### 2) PSD 与 JPG 文件打开后的效果

使用 Photoshop 软件打开 PSD 和 JPG/PNG 图时,“图层”面板的表现有所不同。PSD 图中,原来设计师设计时创建的图层都会被保留;使用 Photoshop 软件打开 JPG 或 PNG 等图像之后,“图层”面板当中只存在一个图层。

#### 2. 找到“被切图像”所在图层

##### 1) 图层的选择

打开“图层”面板(如果找不到“图层”面板,可以通过“窗口”菜单把“图层”面板调出来),使用鼠标单击图层面板中的相应图层,即可选中相应图层;

按住 Ctrl 键,并用鼠标单击相应图层,能够选中多个不连续的图层;

按住 Shift 键,并用鼠标单击两个图层,能够选中这两个图层间的所有图层(含这两个图层)。

##### 2) 图层的快速选择

通过翻阅“图层”面板寻找图像,相对效率较低,此时可以采用更快捷的方法:使用工具





栏中的“选择工具”(位于工具栏第一排的黑色箭头),之后在工具栏菜单当中选择“自动选择”图层/组,之后再使用鼠标单击图像,就能够快速定位该图像的图层所在,如图 1.4 所示。

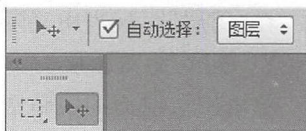


图 1.4 自动选择工具

需要注意的一点是,“自动选择”虽然能够快速地找到某一个图像位置,但是在移动某个图层时,建议及时“取消自动选择”,防止不小心选中了其他图层。

### 3) 查看图层

对于图像,有可能会看不清,可以使用 Ctrl+“+”与 Ctrl+“-”键,实现界面的放大和缩小。如果不想使用快捷键,也可以通过放大缩小工具或百分比数值进行图像显示大小的设置。

图 1.5 中,左上角的为放大镜工具,左下角是图像的缩放比例(建议使用快捷键进行图像的缩放,操作起来会更方便快捷)。

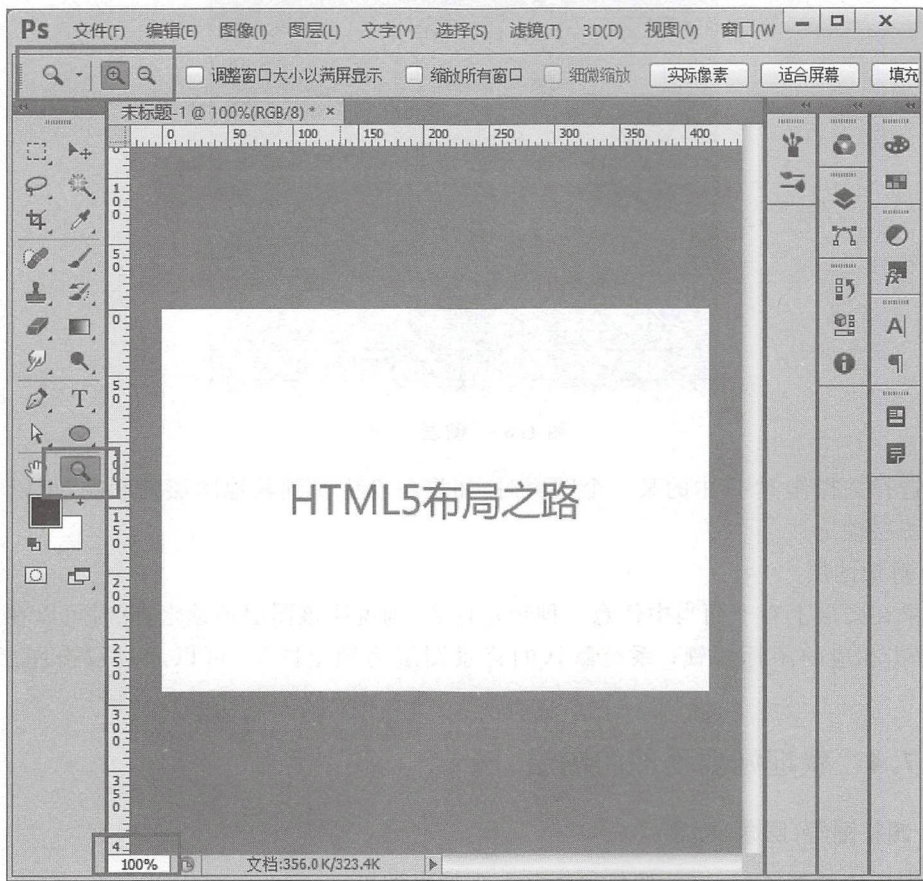


图 1.5 Photoshop 查看图层功能

### 4) 图像的拖曳

当图像大小超出了 Photoshop 显示界面大小时,需要通过拖曳进行查看。此时,不论当前处于任何工具情况下,只需要按住空格键,即可切换到拖曳图像的状态,松开空格键即恢复之前的工具。



### 3. “图层”面板中的图层操作

在“图层”面板当中,能够进行某个或某些图层的“显示”、“隐藏”、“锁定”操作。

#### 1) 显示/隐藏图层

单击图层前面的眼睛图标,就能够实现图层的隐藏,隐藏后单击则为显示图层。

#### 2) 锁定图层

选中图层之后,选择 4 种锁定方式中的一种,进行图层锁定。在图 1.6 当中,从左到右(小方框到锁)分别表示:锁定透明像素(即透明度不能够发生变化)、锁定图像像素(即图像不能够被改变)、锁定位置(即不能够发生运动)、锁定全部(所有的均不能够改变)。

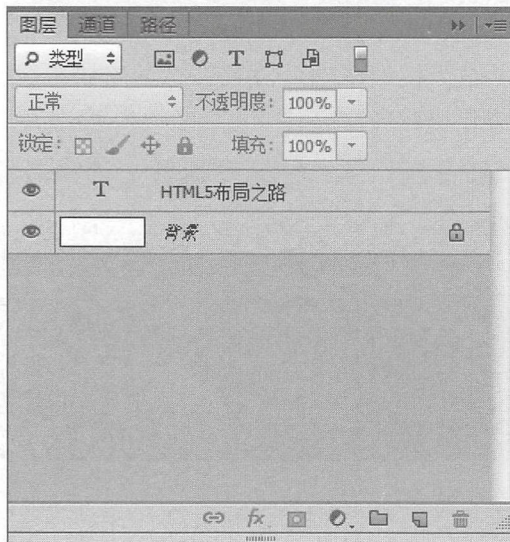


图 1.6 “图层”面板

在选择并操作 PSD 中的某一个图层时,如果不希望受到其他图层的影响,可以将其他不用的图层锁定。

#### 3) 解锁图层

如果在图层上有上面当中任意一种锁定标志,则说明该图层被锁定,此时可以在选中图层后,单击相应的图标解锁;系统默认的背景图层为锁定图层,可以通过双击图层,选择解锁。

## 1.7.4 整理好要处理的图层

### 1. 调整图层(层叠)顺序

选中要调整的图层,之后按住鼠标并上下拖曳即可。

### 2. 图层的合并

选中相应图层后,右键单击,在菜单中选择“合并图层”命令。

### 3. 图层的编组

在选中需要编组的图层之后,选择菜单栏中的“图层”→“图层编组”,也可以使用 Ctrl+G





快捷键。

## 1.7.5 新建文件存储目标图像

### 1. 测量图层大小

#### 1) 使用选框工具进行测量

按 **Ctrl+R** 组合键,调出标尺工具,在标尺上单击鼠标右键,修改单位为“像素”。在标尺上按下鼠标左键,向内容区拖动,能够拉出一条参考线,参考线的作用在于降低量取图层大小时的误差;在参考线上按住鼠标左键进行拖动,可以移动参考线;对于不需要的参考线,可以在参考线上按住鼠标左键,拖动到显示区外部,即可删除。可以通过按 **Ctrl+H** 键,显示或隐藏已经创建的参考线。

在拉好参考线之后,使用“矩形选框工具”贴着参考线创建一个选区,在信息面板当中就会显示这个区域的具体尺寸,如图 1.7 所示。

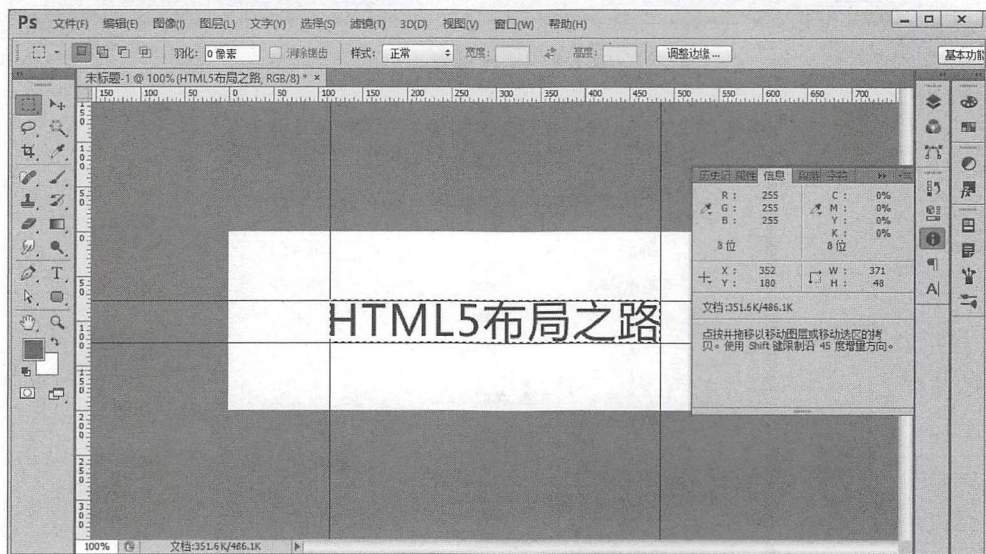


图 1.7 选框工具测量大小

查看大小之后,可以按 **Ctrl+D** 键取消选区。

#### 2) 快捷测量

在“图层”面板当中,找到要测量的图层,之后按住 **Ctrl** 键,用鼠标左键单击图层的缩略图,可以将该图层的内容选中,之后通过“信息”面板进行查看,如图 1.8 所示。额外注意,这种方法并不能够同时操作多个图层。

### 2. 新建文件

在测量完毕图像大小之后,就需要将目标图像(可能是一个图层,也可能是多个图层)从 PSD 图中分离出来,放置于一个新的文件当中,此时需要进行文件的新建。

执行菜单栏中的“文件”→“新建”命令,即可打开“新建”对话框,如图 1.9 所示。

也可使用快捷键 **Ctrl+N**。



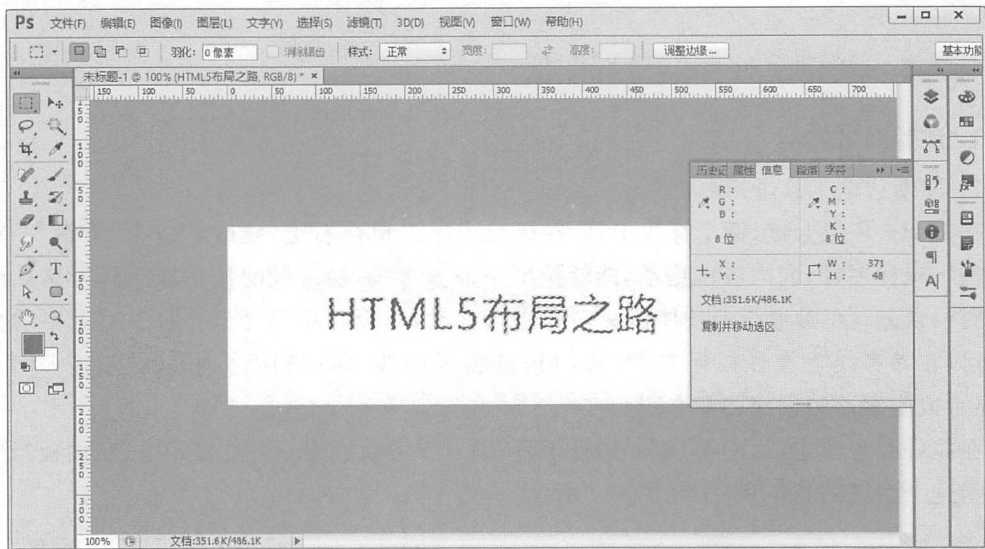


图 1.8 快捷测量大小

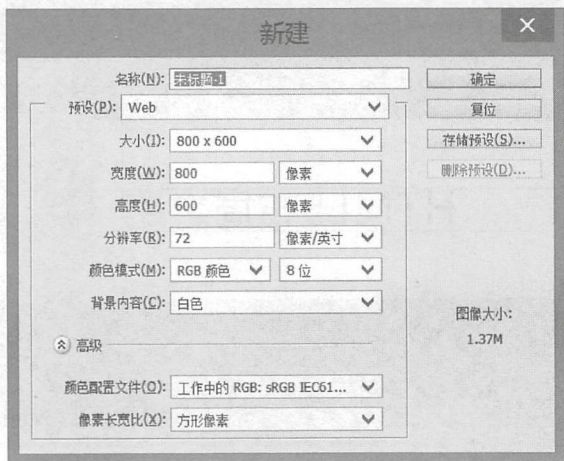


图 1.9 “新建”对话框

### 1) 预设部分

PS 软件为使用者提供了一些常见场景,比如国际标准的 A4 纸张大小,主要目的在于让设计者可以进行更快捷的选择。对于前端来说,通常并不需要创建设计类的文件,可以选择“预设”中的“自定”,来自行定义新文件的宽高及分辨率等信息。另外,在新建文件之后,如果想改变宽度和高度也是没有任何问题的,这一个操作在后面的“修改画布大小”中会提到。

### 2) 宽度和高度

在这个地方请注意单位,选择“像素”。

### 3) 分辨率

不同的设备对分辨率有着不同的要求,网页的分辨率为 72 像素/英寸,而海报等纸质媒





介的分辨率为 300 像素/英寸。此处选择 72 像素/英寸。

#### 4) 颜色模式

在网页当中采用的模式都是 RGB 模式,RGB 表示的是红绿蓝。在默认情况下,RGB 模式的 8 位/通道就足够使用了。但是,具体选择几位通道,还需要查看原设计稿。由于之后要将原 PSD 设计稿中的内容移动到新文件当中,因此,新文件的模式应当与原文件保持一致。通过菜单栏中的“图像”→“模式”,即可查看或修改当前文件模式。

#### 5) 背景内容

如果进行背景图合并,或者不希望背景颜色影响当前的图片,选择“透明”即可。

### 1.7.6 在新文件中调整图像位置

#### 1. 移动内容到新文件

将图层或组(如果进行了编组)选中之后,使用移动工具,将其从原有文件拖动到新文件。

#### 2. 对齐图层

选择多个图层,然后使用移动工具,在移动工具子工具栏当中,能够进行图层的对齐操作。

在子工具栏中,当选择两个或两个以上图层时,如图 1.10 所示,标注 1 中的 6 个功能会被激活,用于实现图层的快速对齐。从左到右依次是:“顶端对齐”,“垂直居中”,“底端对齐”,“左对齐”,“水平居中”,“右对齐”。

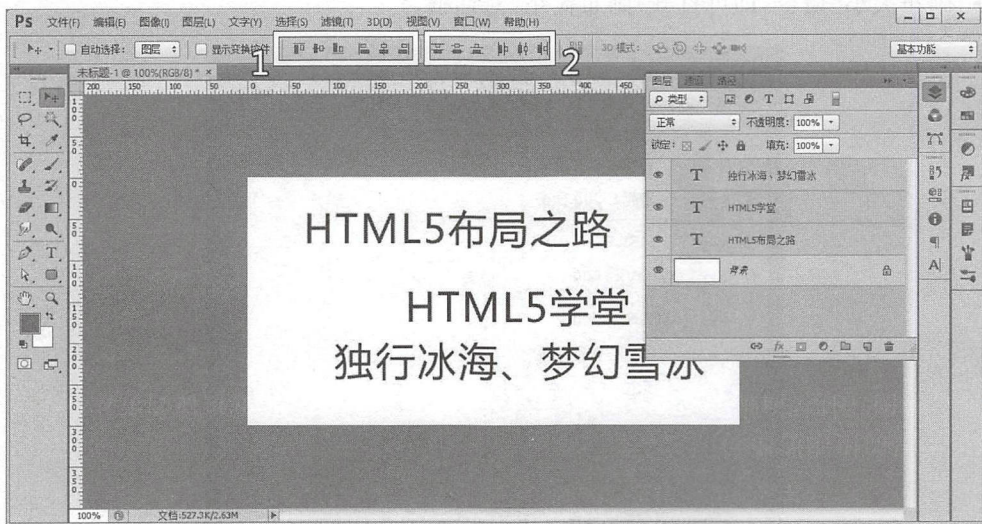


图 1.10 对齐图层的操作

当选择三个或三个以上图层时,图 1.10 中标注 2 中的 6 个功能会被激活,用于调整图层之间的间隔。从左到右依次是:“按顶分布”,“垂直居中分布”,“按底分布”,“按左分布”,“水平居中分布”,“按右分布”。

图 1.10 中的三个图层,在单击“左对齐”和“垂直居中分布”之后的显示效果如图 1.11 所示。



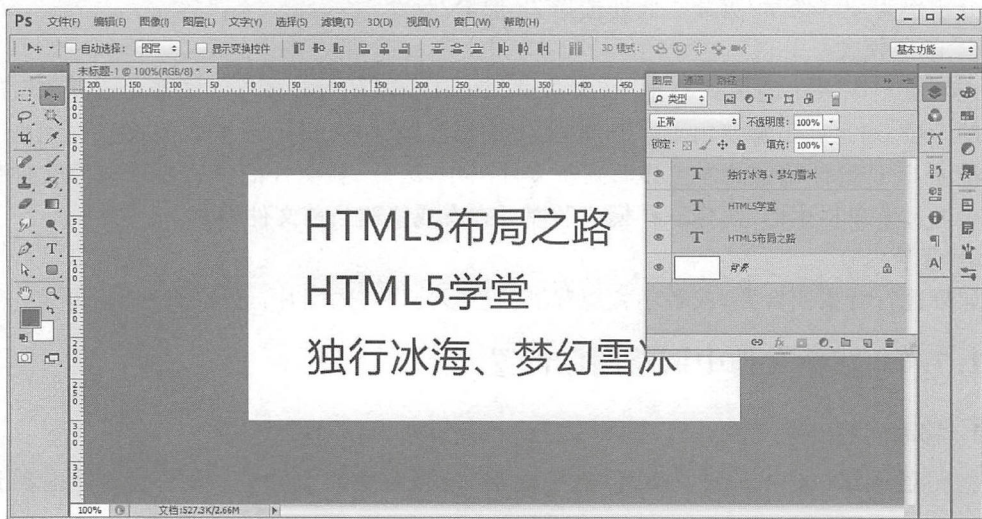


图 1.11 对齐图层操作后的效果

### 1.7.7 修改画布大小

如果在创建 PSD 图时像素大小出现了错误,内容没有办法显示完整或内容占据的空间较少,怎么办?

即便是创建文件之后,依旧可以修改画布的大小。如图 1.12 所示,选择菜单栏中的“图像”→“画布大小”命令,即可打开“画布大小”对话框。

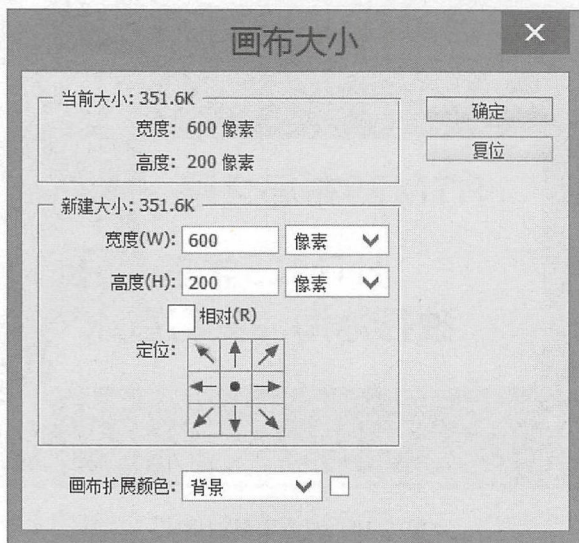


图 1.12 画布大小的处理

在“画布大小”对话框当中,定位指的是“当前的画布”处于哪个位置。假设初始大小为 600 像素×200 像素,想让画布变成 800 像素×400 像素,如果选择中心定位(图中的选择),则画布会从当前位置向四周延伸;如果选择左上角定位,画布则只会向右向下延伸。画布





可以从小到大调整,也可以从大到小调整,但是从大到小的变化有可能会使一些内容从画布中切掉(内容还在,但是由于画布变小,导致内容处于画布显示区域之外)。

### 1.7.8 将图片按照最佳格式类型进行存储

#### 1. 不同的图片格式类型

在网络中使用的图片格式主要有三种,分别是 GIF、JPG、PNG。这三种格式的图片各自有其不同的功能和特点,对动画、透明度的支持以及色彩处理都不尽相同。对于前端来说,在不同场合下要选用合适的图片格式。

如果对这些图片格式的了解一无所知,只是单纯地追求高清晰度的图片而乱用的话,或许会大大降低网站的性能,得不偿失。下面介绍图片的各种格式类型。

##### 1) GIF 图片格式

(1) GIF 格式的图片支持全透明,它仅可以是完全透明或完全不透明,并不支持半透明;

(2) GIF 格式的图片相对图像质量较差;

(3) GIF 支持动画,一个 GIF 文件中可以存储多幅彩色图像,如果把存于一个文件中的多幅图像数据逐幅读出并显示到屏幕上,就可构成一种最简单的动画。

##### 2) JPG 图片格式

(1) JPG 格式的图片不支持任何透明的显示,因此当图片有透明度要求的时候,可以先放弃 JPG 格式;

(2) JPG 支持最高程度的压缩(有损压缩),当需要没有透明要求的大图时,可以尝试保存成 JPG 格式,适当地压缩到人眼舒适的程度,色彩质量相对较好。

##### 3) PNG 图片格式

(1) PNG 图片又可以划分为 PNG8、PNG24 和 PNG32,在平时的工作中,PNG24 已经足够使用了。和 GIF、JPG 格式相比,PNG 格式流行的比较晚,浏览器和相关图片处理软件对 PNG 格式图片的处理也是随着其发展逐步成熟的;

(2) PNG 格式支持半透明显示,还支持真彩和灰度级图像的 Alpha 通道透明度,PNG 图片能获得高的压缩比而不损失数据(也称无损压缩);

(3) IE6 只支持 PNG8 格式的图片,PNG24 格式的图片在 IE6 浏览器下,透明部分会变成灰色。

#### 2. 存储文件

执行菜单栏中的“文件”→“存储”命令,即可打开“存储为”对话框,如图 1.13 所示。也可以使用快捷键 Ctrl+S。

需要注意的是,如果当前操作文件为 PSD 类型,且文件从来没有保存过,则会弹出“存储为”对话框。如果文件操作过,则不会弹出任何对话框,而是默认覆盖当前 PSD 文件。如果当前操作的文件是非 PSD 类型的文件,操作过程中没有出现多个图层,则存储命令为覆盖当前文件;如果操作中出现了多个图层,会弹出“存储为”对话框。

#### 3. 存储为其他格式

除了能够保存 PSD 类型之外,也可以将当前文件保存成自己想要的其他格式,有以下



图 1.13 “存储为”对话框

两种方法。

方法 1：使用“存储为”命令。

执行菜单栏中的“文件”→“存储为”命令，即可打开相应对话框。

快捷键：Shift+Ctrl+S。

方法 2：存储为 Web 所用格式。

执行菜单栏中的“文件”→“存储为 Web 所用格式”命令，即可打开相应对话框，如图 1.14 所示。

快捷键：Shift+Ctrl+Alt+S。

与方法 1 相比，方法 2 操作起来相对比较复杂，功能也更加强大，能够进行不同类型文件的预览图对比，能够进行更详细的设置，还能够配合切片工具使用。

#### 4. 将多个背景图一次性存储下来——切片工具

切片工具的主要作用，在于将多张大小类似的图片从一张图像中同时切割，提升切图效率。由于切片工具操作较为复杂，在此以一个实例进行讲解。

1) 根据需求，进行内容排布

需求：该图为 300 像素×200 像素的画布，共包含 6 个小图标，当前希望将每个小图标切出来，每个图标大小为 100 像素×100 像素，如图 1.15 所示。



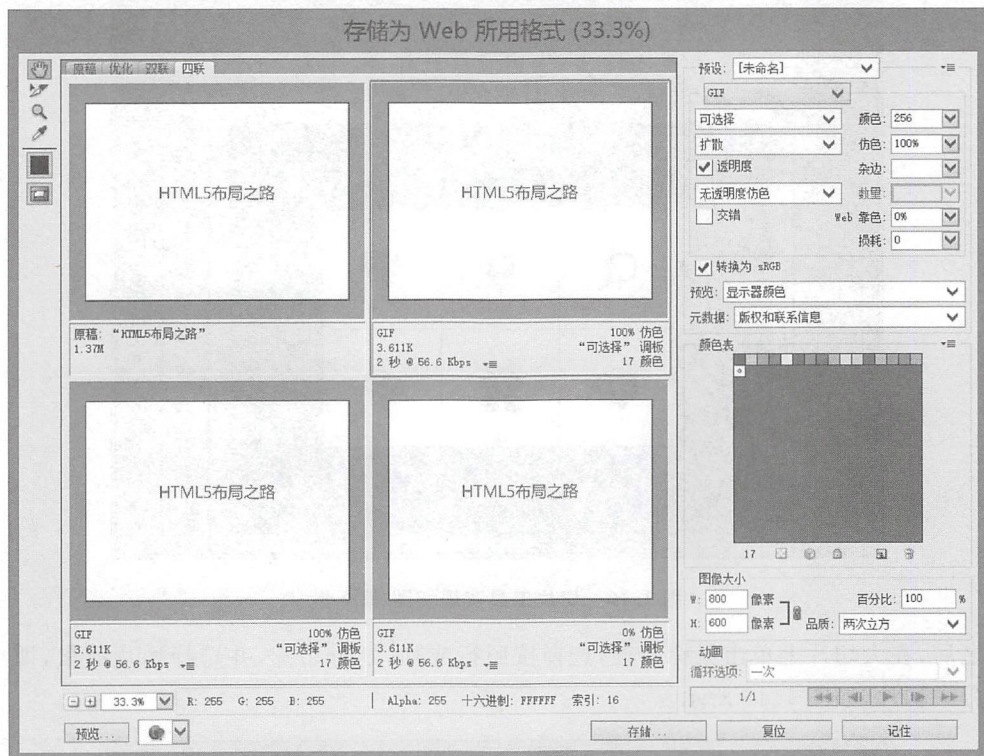


图 1.14 存储为 Web 所用格式



图 1.15 切片工具实现切图-第1步

根据具体需求,使用选择工具进行图标的排布。排布效果如图 1.16 所示。

## 2) 使用切片工具,创建切片

先使用切片工具选中整个图片(在显示区域左上角按住鼠标,拖曳到右下角,保证选框能够包含整个显示区域),使整个显示区域构成一个大的切片。

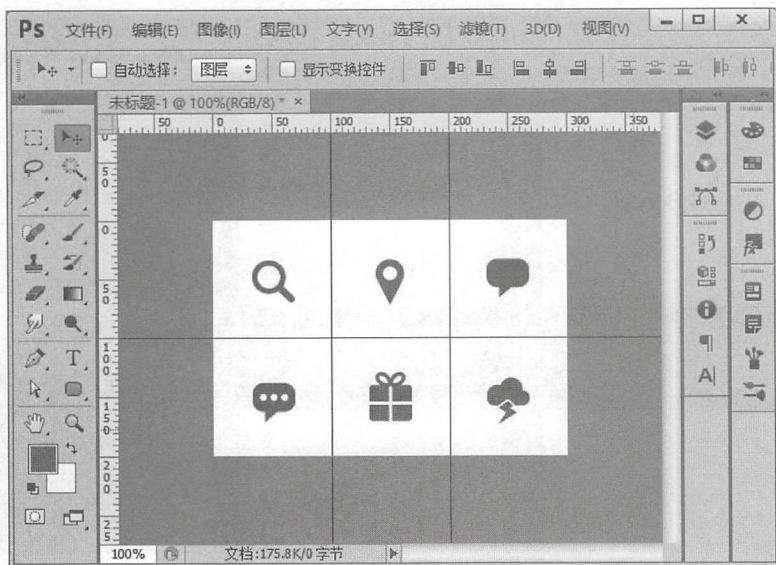


图 1.16 切片工具实现切图-第 2 步

之后,在大切片上单击鼠标右键,选择按照行或列,划分切片,并选择切片数量,即可自动平分大切片,如图 1.17 所示。



图 1.17 切片工具实现切图-第 3 步

### 3) 存储切片

创建完成切片之后,执行“文件”→“存储为 Web 所用格式”命令,为每个切片选择需要存储的格式。之后单击“存储”按钮,在弹出的“将优化结果存储为”对话框当中选择“所有切片”,如图 1.18 所示。







## 1.8 使用 Photoshop 获取图层信息

### 1.8.1 获取颜色

#### 1. RGB 颜色

RGB 即红绿蓝,称为色光三原色。这个标准几乎包括人类视力所能感知的所有颜色,是目前运用最广的颜色系统之一。而与计算机的相关之处,就在于屏幕上的所有颜色,都由这三种色光按照不同的比例混合而成,屏幕上的任何一个颜色都可以由一组 RGB 值来记录和表达。

RGB 中的每一种颜色,都可以用整数来表达,每种颜色各有 256 级亮度,分别用 0~255 来表示,0 表示最暗,255 表示最亮。

在 Photoshop 当中获取的颜色可以直接应用于网页开发当中,并且 Photoshop 也提供了多种颜色的表示方法。一种是 rgb 的数值,另一种是十六进制表示法(色值)。例如,红(纯)色可以表示为 rgb(255, 0, 0),也可以表示为 #ff0000。

在网页的色彩表示当中,除了上面的两种方法之外,还可以使用具体的英文单词来表示颜色,如红色可以使用 red 来表示。另外,对于 #ff0000 此类表示法,在一定情况下可以进行缩写,缩写要求为红、绿、蓝三种,各自的两个数值均相同(如 #3399ff 可以缩写为 #39f),如果有任意一个不同,就不能够缩写(如 #39ff00 就不能够缩写)。

- 扩展知识:不同颜色混合是什么颜色

(等量的)红+绿=黄;绿+蓝=青;红+蓝=品(紫/品红)

对于颜色的混合,可以采用彩虹记忆法:红(橙)黄绿青蓝紫。

白色为 #ffffff,对应 rgb(255, 255, 255);黑色为 #000000,对应 rgb(0, 0, 0)。

- 扩展知识:CMYK 印刷色彩

CMYK 是专门针对印刷业设定的颜色标准,是通过青(C)、洋红(M)、黄(Y)、黑(K) 4 个颜色的变化以及它们相互之间的叠加来得到各种颜色的。CMYK 即是代表青、洋红、黄、黑 4 种印刷专用的油墨颜色,也是 Photoshop 软件中 4 个通道的颜色。

CMYK 是以对光线的反射原理来设计指定的,所以它的混合方式刚好与 RGB 相反,是“减法混合”,当它们的色彩相互叠合的时候,色彩相混,而亮度却会减低。

#### 2. 关于 alpha 透明度

每个图层的透明度,可以通过“图层”面板中的“不透明度”和“填充”来查看。

“不透明度”控制的是整个图层的透明度,图层样式也会跟着变化;“填充”控制的是图层内容的透明度,图层的描边等图层样式(fx)并不会随之变化(设计师在开发时会选择不同的方式来控制透明度,因此,此处需要知晓“不透明度”和“填充”的区别)。

#### 3. 获取颜色的方法

通用但不精确的方法:使用吸管工具。选择吸管工具,在需要测量的颜色位置单击左键(吸取颜色),之后前景色会变成这个颜色,单击前景色部分,就能够查看具体颜色对应的色值了。



精确获取颜色的方法,因内容不同而异。

(1) 对于路径,采用双击图层缩略图右下角的路径的方法,会弹出“拾色器”对话框,如图 1.20 所示。



图 1.20 获取路径的颜色

(2) 对于图层,有些会有特殊样式(标有 fx 的),双击图层右侧的 fx,打开“图层样式”对话框,再进行查看,如图 1.21 所示。

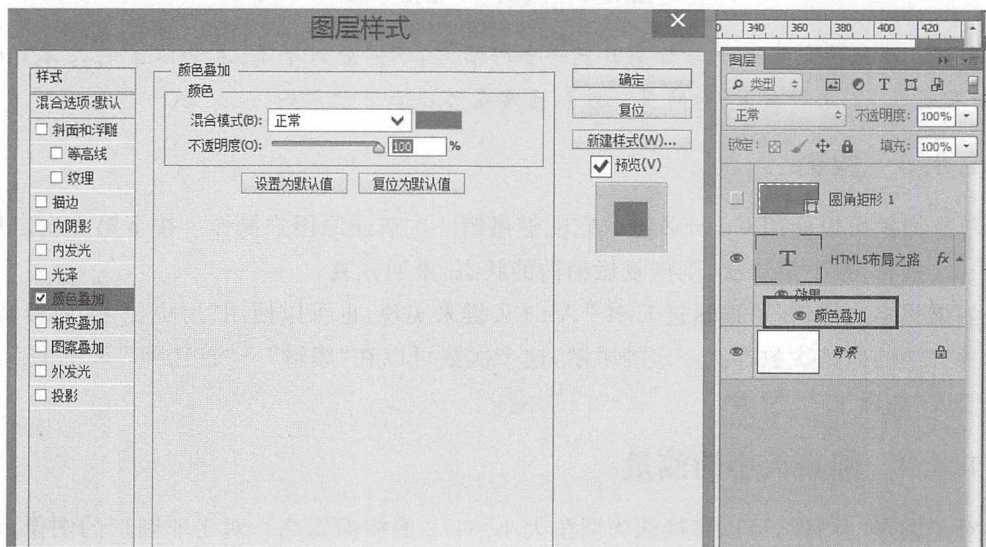


图 1.21 获取图层特殊样式的颜色

(3) 对于字体颜色,详见 1.8.2 节的“文字内容获取”中的讲解。

### 1.8.2 文字内容与特点

使用工具栏中的文字工具,在相应文字图层上单击,即可获取文字图层的内容。

对于文字的一些特点,可以在选中文字图层之后,通过“字符”与“段落”面板进行查看。主要包括字体类型、字体是否加粗、倾斜、下画线、字间距是否进行过调整、字体颜色、文字大小与行高,如图 1.22 所示。

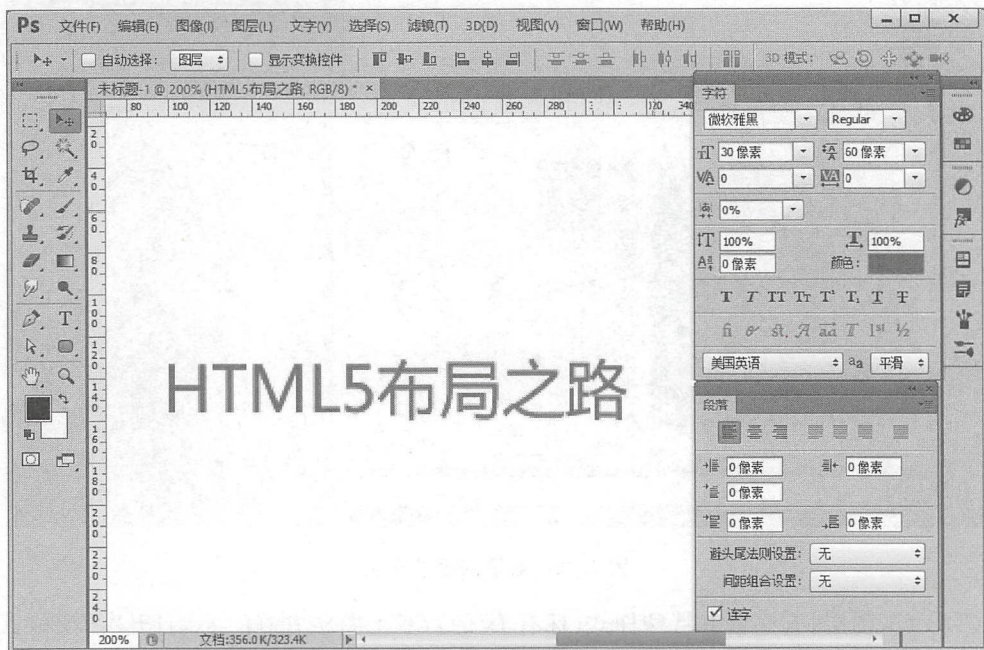


图 1.22 “字符”与“段落”面板

**备注:** 此处的文字大小与行高只能够当作参考,不同分辨率下,点和像素的换算有所不同,如果是在 72 分辨率下(和网页一致),通常是可以正常使用的。

### 1.8.3 撤销

单次撤销可以使用 **Ctrl+Z** 键。它能够撤销一次或还原用户操作。按下第一次之后为撤销一次操作,按下第二次,则恢复撤销前的状态,来回往复。

如果想多次撤销,可以通过 **Ctrl+Alt+Z** 键来实现,也可以使用“历史记录”面板实现。这两种方法的撤销次数都有一定的限制,这个次数可以在“编辑”→“首选项”→“性能”当中进行修改,如图 1.23 所示。

### 1.8.4 圆角大小的测量

对于圆形的图像,可以直接认为圆角大小=1/2 的图像宽高。对于非圆形的图像,可以用如下两种操作方法。

**方法一:** 由于设置圆角的图像,在边界部分,均采用的是半透明颜色。因此,可以不断放大,直到能够清晰查看每个像素点,之后,从元素左上角开始,选择到圆角弧度消失(颜色





为实际颜色)为止即可,如图 1.24 所示。这种方法对视觉的敏锐度要求很高,通常靠近边上的色块会很接近实际颜色,但是并非就是实际颜色。

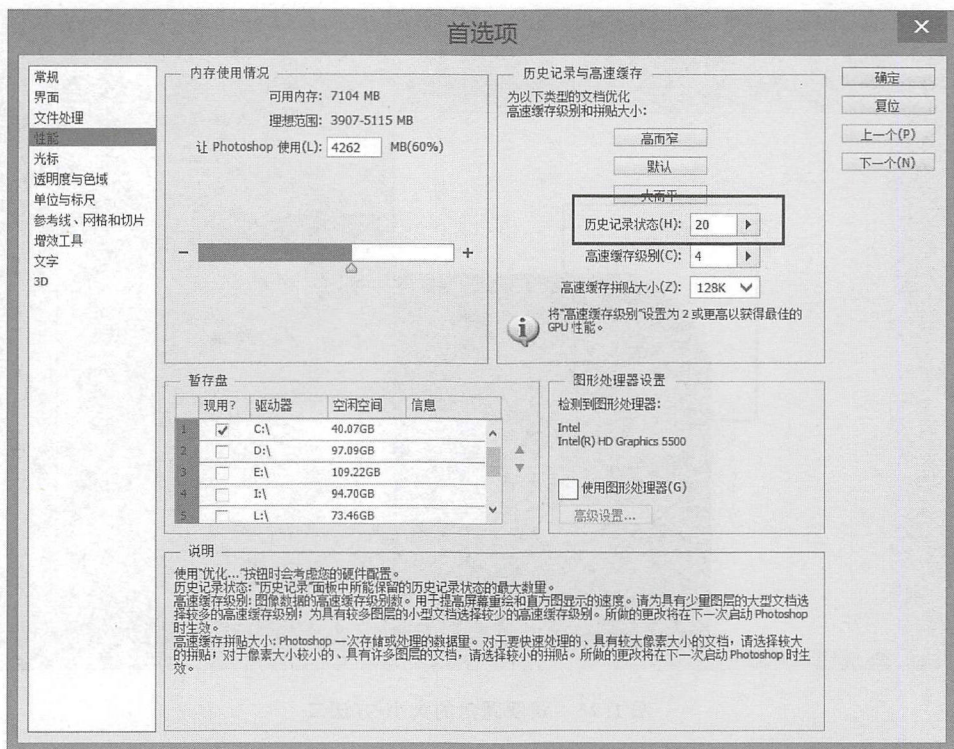


图 1.23 设置历史记录状态

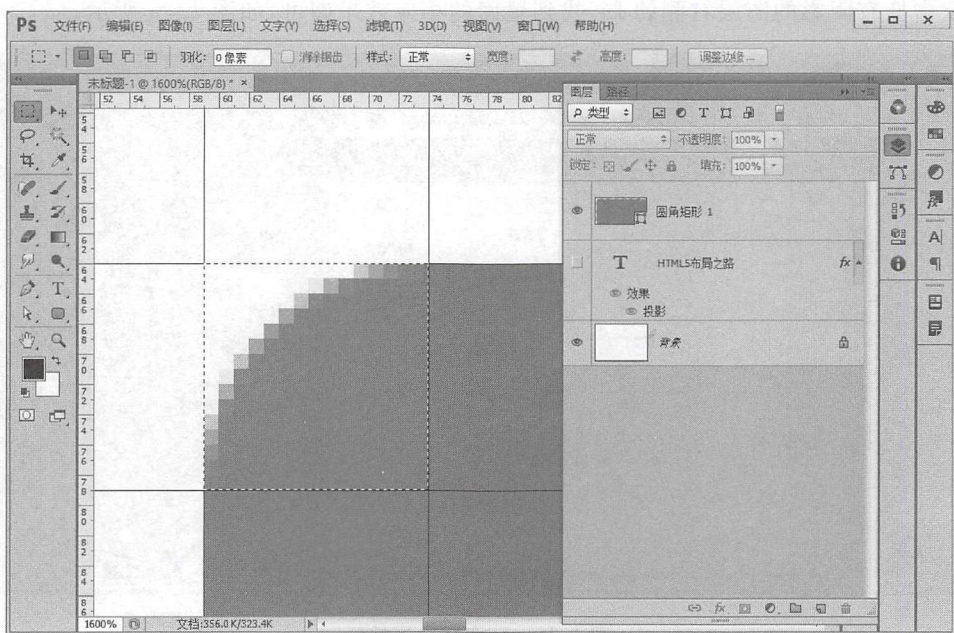


图 1.24 获取圆角的大小-方法一



## HTML5布局之路

方法二：预估一个值，采用路径工具在相同位置绘制一个，之后调整上面的圆角值，保证和原有图贴合或路径的弧度相同，如图 1.25 所示。

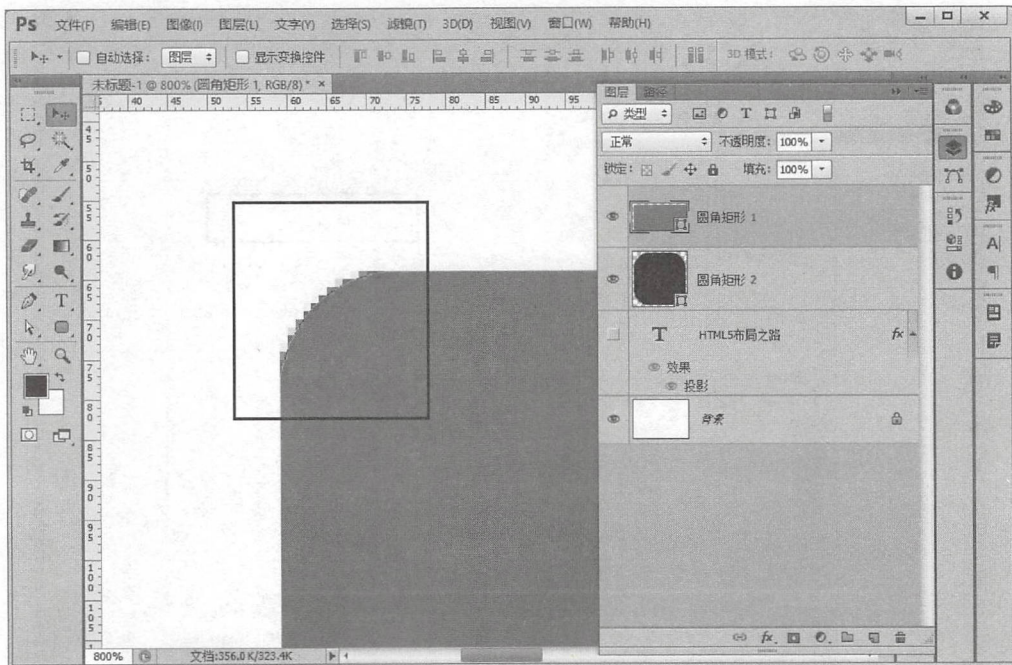


图 1.25 获取圆角的大小-方法二

### 1.8.5 阴影的测量

双击拥有阴影的图层右侧的 fx，找到里面的相应效果即可，如图 1.26 所示。

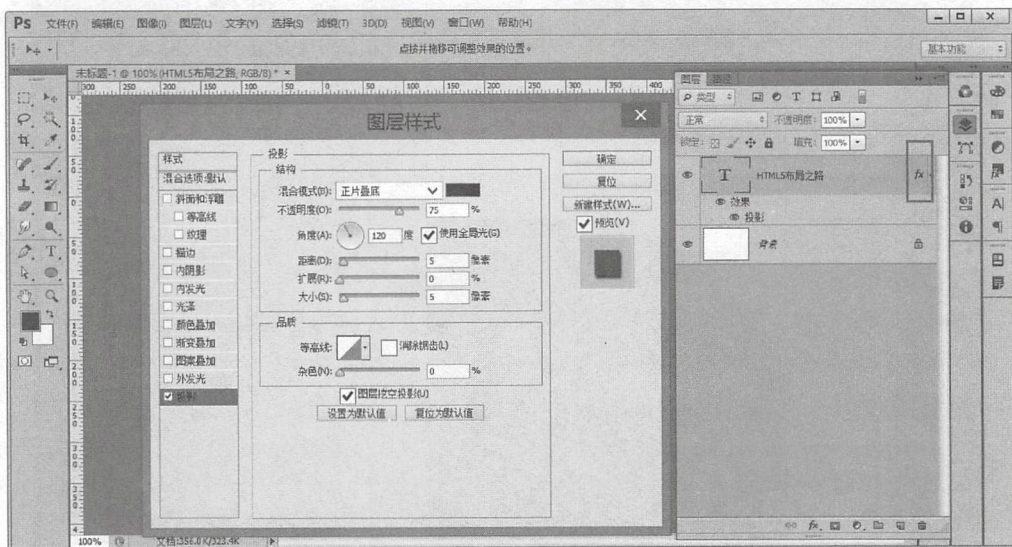


图 1.26 获取阴影属性值





### 1.8.6 将文字处理成图像

在“图层”面板当中,选中文字图层,在该图层上面单击鼠标右键,选择“栅格化”命令即可。

### 1.8.7 Photoshop 快捷键总结

Photoshop 中各类快捷键如表 1.2~表 1.5 所示。

表 1.2 各类软件通用快捷键

快 捷 键	快捷键功能
Ctrl+N	新建
Ctrl+O	打开
Ctrl+W	关闭
Ctrl+C	复制
Ctrl+V	粘贴
Ctrl+X	剪切
Ctrl+S	默认保存

表 1.3 Photoshop 测绘用快捷键

快 捷 键	快捷键功能
Ctrl + +	新建放大
Ctrl + -	打开缩小
Ctrl+R	关闭隐藏/显示标尺
Ctrl+H	复制隐藏/显示辅助线

表 1.4 撤销与另存快捷键

快 捷 键	快捷键功能
Ctrl+Z	一次撤销/还原
Ctrl+Alt+Z	多次撤销
Shift+Ctrl+S	另存为
Shift+Ctrl+Alt+S	存储为 Web 常用格式

表 1.5 其他快捷键

快 捷 键	快捷键功能
Ctrl+D	可以取消选区
Ctrl+T	让某个图层图像变形
Ctrl+G	编组
Ctrl+Alt+G	取消编组



## 1.9 代码编辑器

HTML 代码是一种超文本标记语言,通俗地说,就是一些浏览器能读懂的文本,它和 txt 文本中编辑的内容类似,只是它有自己的基本语法和要求。

代码编辑器,能够契合 HTML5 的基本语法,提升开发者的开发效率。在行业当中有很多种编辑器,它们的原理与基本操作方法相同,在此以 Sublime Text 编辑器为例讲解编辑器的使用与相关操作,之后给出常用的其他编辑器,可以根据自己的情况进行选择。

### 1.9.1 Sublime Text

#### 1. Sublime Text 操作界面

Sublime 界面如图 1.27 所示。

(1) 在百度当中直接搜索“Sublime Text”即可找到免费的下载地址,有 Sublime Text 2 和 3 两种版本,均可下载。

(2) 所有的操作都可以在顶部菜单栏当中找到,翻看一下菜单栏就能够快速了解软件的所有功能。



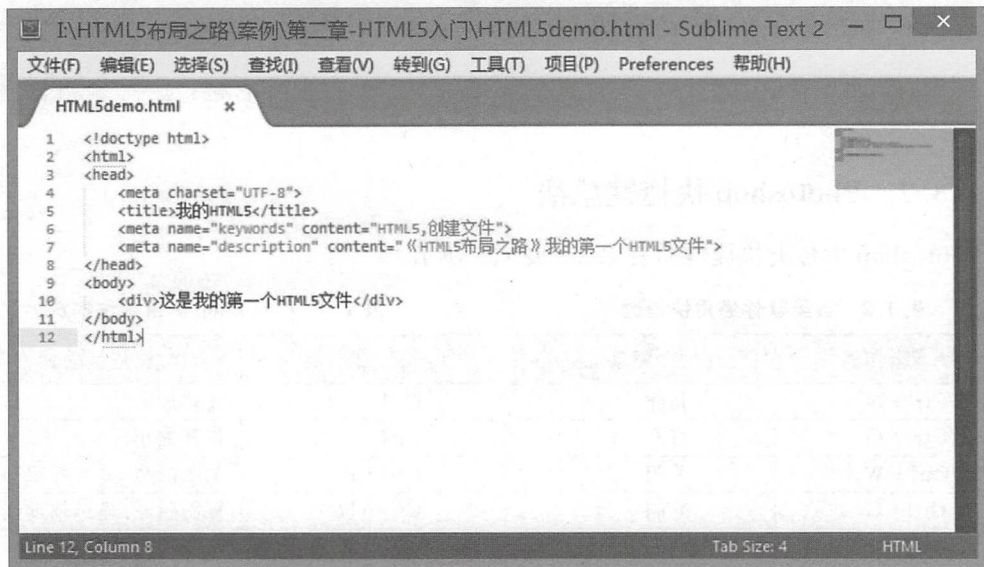


图 1.27 Sublime 界面

(3) 在编辑器当中能够打开多个文件,每个文件都会以一个选项卡的方式打开。

(4) 内容区域当中,会有不同的配色,不同代码不同颜色,一方面缓解了视觉疲劳,另一方面(也是最重要的)能够让开发者快速区分内容类型。

(5) 代码的左侧是行数,更方便进行阅读和查找。

(6) 代码右侧为代码的“迷你地图”,当页面代码非常多时,可以通过拖曳迷你地图,快速上下移动从而查看代码,可以通过“查看”菜单来控制显示或隐藏迷你地图。

## 2. 编辑器的汉化

有可能下载的 Sublime 编辑器是英文版,如果希望转换为中文版,可以进行编辑器的汉化,具体步骤如下。

(1) 下载 Sublime 汉化包(可以在百度中查找,也可以使用本书电子资料中提供的汉化包)。

(2) 解压汉化包,将 default 文件夹中的 8 个文件取出。

(3) 打开 Sublime,找到菜单中的 Preferences→Browse Packages 命令。打开文件夹之后,找到 default 文件夹,然后将刚才取出的 8 个文件替换掉原来的文件即可。

## 3. 编辑器的快捷键

每款编辑器都有一系列的快捷键,熟练掌握快捷键能够有效地提升开发者的编码速度。在此罗列一下 Sublime 常用的快捷键(对于剪切、复制、粘贴、打开、关闭、新建等通用型快捷键,在此就不做介绍了),如表 1.6 所示。

## 4. 编辑器插件

如果说编辑器是为了提升开发者的编码速度,那么编辑器插件就是让开发者的编码速度更上一层楼。不过对于初学者来说,刚开始时并不建议使用插件。作为一个编写代码的工程师,必须要了解基本的代码书写规范,并能够在书写过程中记忆清晰,对于开发中使用





表 1.6 Sublime 常用快捷键

快 捷 键	功 能 含 义
Tab	缩进一次(可以选择多行时按下,所有被选中行均会被缩进)
Shift+Tab	反向缩进一次(可以选择多行时按下,所有被选中行均会反向缩进)
Ctrl+/	注释/解开注释
Ctrl+Z	撤销操作
Ctrl+L	选择当前行,如果多次连续按下该快捷键,会继续选择下一行
Shift+Ctrl+上/下	移动代码行位置
Ctrl+G	定位行号
Ctrl+H	替换
Ctrl+F	查找
Shift+Ctrl+<	上一个编辑点
Shift+Ctrl+>	下一个编辑点
Shift+Ctrl+[	折叠代码
Shift+Ctrl+]	打开折叠
Alt+数字	对应第几个选项卡文件(即 Alt+1,就是切换到第一个)

到的一些单词也是必须要记下来的。代码质量和代码速度其实都是开发者应当追求的,建议在对基础知识扎实掌握之后,再使用插件,将自己的编码速度提升一个档次。

对于具体插件的安装以及应用方法,详见附录。

## 1.9.2 其他代码编辑器

### 1. Webstorm

Webstorm 近来被行业热炒,被开发者誉为“Web 前端开发神器”、“最强大的 HTML5 编辑器”。因其强大的智能提示功能而出名(Sublime 没有智能提示),能够进行代码补全、代码检测、快速修改、代码智能重构等,让代码简洁干净安全,工作效率大大提升,适用于实际的开发工作人员。支持 Windows、Mac OS 和 Linux 三个主流操作平台。

### 2. Visual Studio

Visual Studio 支持 Windows 开发环境,支持大量的开发语言(如 C/C++、C#、VB.NET 和 F#),可以用来开发桌面应用、移动端和网页。拥有强大的自动补齐,错误效验,表单设计等功能。虽然该编辑器收费,但是其快速版本可免费使用,但免费版的开发特性有所限制。

### 3. HBuilder

HBuilder 的开发效率很高,引入了“快捷键语法”的概念,巧妙地解决了困扰许多开发者的快捷键过多而记不住的问题。开发者只需要记住几条语法,就可以快速实现跳转、转义和其他操作。其软件性能相对较差,占用内存空间比例较高。

### 4. Eclipse

Eclipse 是著名的跨平台的自由集成开发环境(IDE)。最初主要作为 Java 语言的开发工具,但是目前也有人通过插件使其作为 C++、Python、HTML5、PHP 等其他语言的开发



工具。

Eclipse 的本身只是一个框架平台,但是众多插件的支持,使得 Eclipse 拥有较佳的灵活性。

## 5. Dreamweaver

Dreamweaver 属于 Adobe 应用套件之一,主要用来开发 Web 应用。为了方便初学者的编程,支持所见即所得的编辑方式。

当在所见即所得的 Dreamweaver 中制作好的网页,将其放入浏览器中有时很难完全达到满意的效果,这常常在结构较为复杂的一些动态网页结构中很容易显现出来,难以精确达到与浏览器完全一致的显示效果。



## 1.10 浏览器调试

### 1.10.1 为何要进行浏览器调试

进行浏览器调试的目的是让网页在各个浏览器当中都能够表现一致,除了能够正常地呈现信息,供用户使用之外,在样式上、行为上都能够与最初“设计构思”的效果相符合。

在开发当中,开发工程师会遇到如下两种问题,这两种问题也是要求开发工程师及时调试的原因。

#### 1. 浏览器对代码的支持程度不同

有些浏览器支持某些命令,而有些浏览器不支持,这也就导致网页在部分浏览器当中表现正常,在部分浏览器当中表现不正常。通过浏览器调试,可发现这些问题并及时解决。

#### 2. 开发者自身问题

开发工程师在书写代码时,难免会书写出“错误”或“有问题”的代码。例如,子级元素的大小超出了父级元素的大小,此时,代码在容错性比较高的浏览器(如谷歌 Chrome 浏览器)当中并不会出现什么问题,但是在容错性比较差的浏览器(如 IE6~IE8)当中就有可能出现页面布局崩溃的现象。

进行代码的调试,就是为了及时发现问题,并且尽快解决问题,以防止书写了很多代码之后,突然发现在某些浏览器中是错乱的,而原因却又无从查起,最后不得不重新书写或花费大量时间在查找错误上。

### 1.10.2 浏览器调试的基本要求

(1) 随着开发的进行而不断测试,开发一部分就测试一部分,较高的测试频繁度,能够尽早发现问题,随着开发经验的增长,测试频率可以逐渐降低;

(2) 每次测试,都应当测试“需求”中要求兼容的所有浏览器,通常情况下,主流浏览器包括 IE9~IE11、Firefox(火狐)、Chrome(谷歌)、360 等,较低端浏览器的测试当中,还要考虑 IE6~IE8;

(3) 合理地运用谷歌、火狐等浏览器中的控制调试工具,更快捷地找出代码的错误。







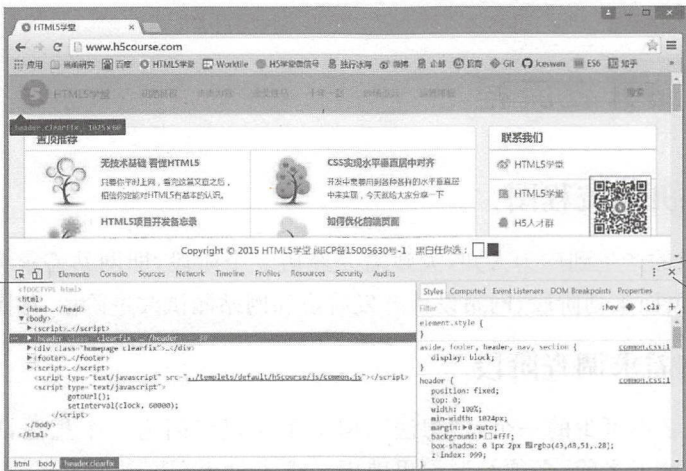
### 1.10.3 浏览器调试方法

#### 1. 控制台调试

在谷歌、火狐、IE9+等各个主流浏览器当中,都为开发者提供了控制台工具,开发者可以在浏览器当中按下 F12 键,即可调出浏览器的控制台。每个浏览器的控制台都是类似的,在此以谷歌浏览器的控制台为例进行讲解。对于其他浏览器的控制台,原理与操作方法基本相同,只是在某些菜单和功能上有所不同,多使用多操作即可掌握。

(1) 打开网页,在谷歌浏览器当中按下 F12 键,呈现如图 1.28 所示效果。

③ HTML与CSS  
调试时使用的  
面板



① 控制台的菜单部分,  
可以进行更多的设  
置与操作。  
② 可以关闭控制台

图 1.28 控制台调试

(2) 控制台右上角有一个“关闭”按钮(×)和三个点。在打开控制台之后,可以通过“关闭”按钮来关闭控制台,也可以再按下 F12 键关闭控制台。三个点的小图标,包含控制台中的更多操作,例如,可以将控制台部分与浏览器窗口分离开,作为两个窗口来查看,操作更加便捷。

(3) 在控制台中,包括 Elements、Console 等各类子面板,其中用于 HTML 与 CSS 调试的面板是 Elements,Console 面板主要用于调试 JavaScript,对于 Sources、Network 等面板,在实际的项目开发当中也会或多或少地使用到。

(4) Elements 面板的用法:控制台中 Elements 子面板,“浏览器中的元素”、“左侧的结构”以及“右侧的 CSS 样式”三者对应,同时变化。

使用鼠标单击控制台左上方的(箭头)小图标,之后可以用鼠标去选择浏览器中的任意部分,被选择到的元素会以高亮状态进行呈现;与此同时,控制面板左侧的结构代码会自动定位到当前选择的标签位置,控制面板右侧的样式代码也会切换为当前元素相对应的样式属性。

如果使用鼠标选择控制面板左侧的结构标签,面板右侧的 CSS 以及浏览器窗口中的内容都会随之改变。如果针对控制面板右侧的 CSS 进行修改操作,浏览器中的元素也会发生变化。

#### 2. 其他调试工具或软件

除了控制台之外,火狐还提供了 FireBug 插件,用于辅助代码调试;如果需要测试 IE6,可以下载 IE Tester 软件,能够模拟 IE6~IE10 的浏览器(由于 IE Tester 软件本身存在一定的问题,有时显示效果并不是很准确)。





## 第2章

# HTML5入门



### 2.1 网站开发流程

#### 2.1.1 网站开发流程图

一个网站从需求的产生到广为人知,大致需要经历 5 个阶段,即网站需求分析阶段、网站技术分析阶段、网站页面策划阶段、网站设计开发阶段和网站测试改进阶段,如图 2.1 所示。

#### 2.1.2 网站需求调查阶段

开发一个网站,必不可少的一个是“想法”,但却不能仅仅因为一个想法,就决定开发建设一个网站,在真正动工之前,必须经过深思熟虑。需求分析在网站开发过程中具有举足轻重的地位,因为它具有目的性、方向性、决策性,在任何网站的开发中,其作用都要远大于设计或编码,原因很简单,方向错误,跑得再快都是徒劳无功。假设你是一家企业的老板,想开发一个网站,那么应对需求分析具有足够的重视。一个网站基本的需求分析,至少需要包含如下几个方面。

##### 1. 功能清晰完善

在开发一个网站时,首先要分析清楚自己要的是什么,需要的网站具有什么功能,能够为哪些用户群体提供哪些类型的服务。简言之,网站定位要清晰。

##### 2. 时间以及财务支出

每个网站(或产品或项目)都存在开发周期(预计投入的时间),以及预计投入的财力,那么就必须考虑当前的需求能否在规定时间内以及财力范围之内完成。

##### 3. 竞争对手

绝大多数网站的目的都是“盈利”,不管是通过网站盈利,还是通过网站宣传公司或商品而盈利。

对于以“网站”作为产品的老板,在产生网站需求之前就一定要清楚自己的“竞争环境”。

例如,一家企业想开发一个搜索引擎类型的网站作为自己的主要业务,经过用户分析,发现这种类型的网站有着大量的使用人群,经过技术以及财务等估算,也能够工期与财务限额之内完成。看上去这个网站是可以开发了。但是,当考虑到后面网站的推广以及让用户使用这款搜索引擎,这个需求就成为泡影,因为做搜索引擎类型的网站,要面对谷歌、百





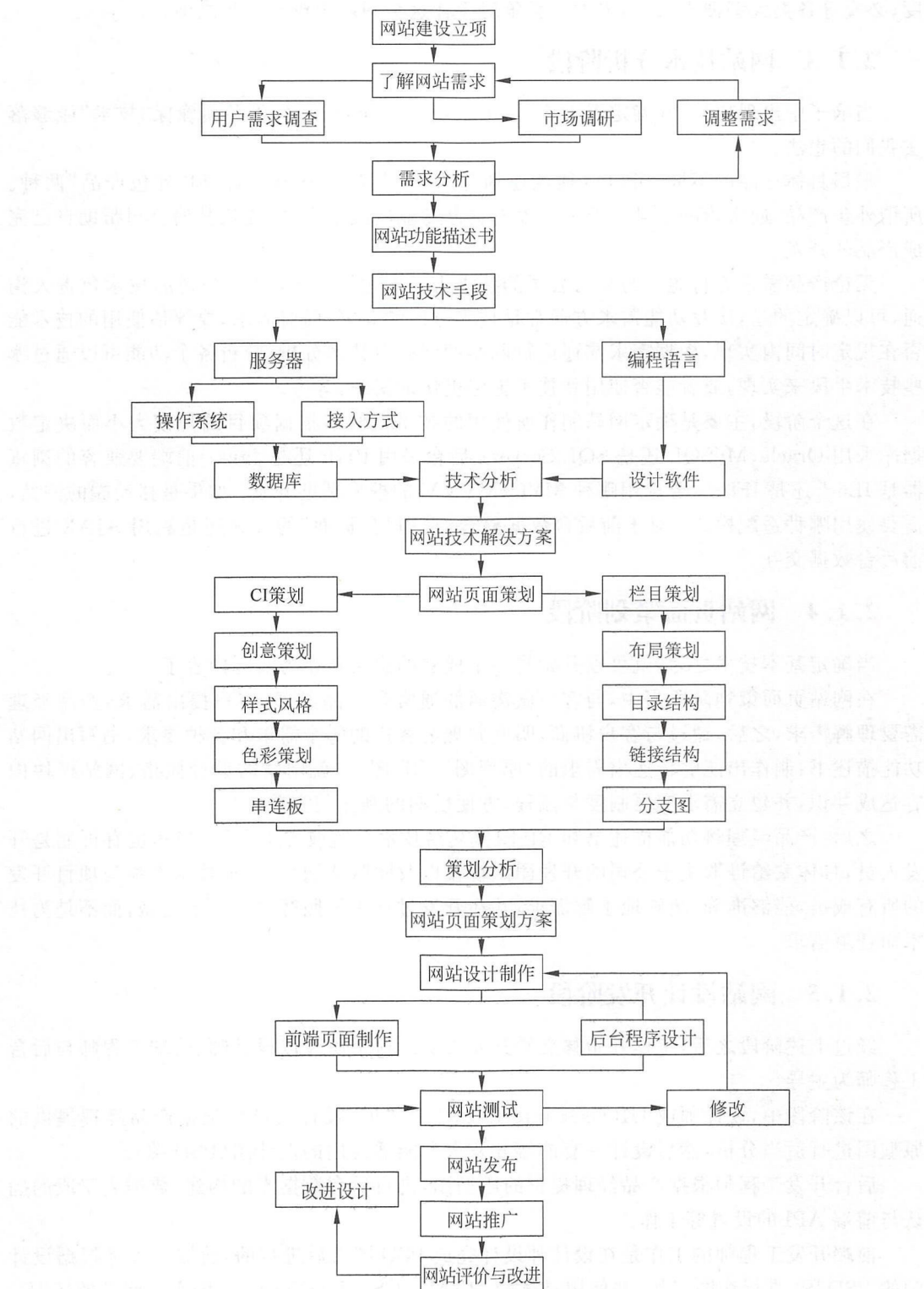


图 2.1 网站设计整体策划流程图



度、必应等各类大型搜索引擎,强大的竞争对手让这个网站很难有立足之地。

### 2.1.3 网站技术分析阶段

当脑子中出现想法,并确定需求之后,也不能立即开始开发,还必须确保“技术”能够落实我们的想法。

根据具体产品的不同,可以概括性地划分为“开发自己公司产品”和“外包产品”两种。所谓外包产品,通常指的是某一家公司没有自己专业的技术人员,委托其他公司帮助自己完成产品的开发。

无论产品需求方将想法与自己公司的技术人员沟通,还是与其他公司的技术负责人沟通,可以确定的是,接收功能需求方通常是技术方面的专家,面对需求,要评估使用的技术能否在规定时间内实现,根据需求描述进行网站的前后台技术分析,分析各个功能可以通过哪些技术手段来实现,或者能否使用新技术更好更快地实现,等等。

在这个阶段,主要是确定网站制作所使用的技术,例如,根据项目的规模大小而决定数据库采用 Oracle、MySQL 还是 SQL Server; 后台采用 PHP 还是 Java; 前端要兼容的浏览器是 IE6+ 还是 IE9+, 要选用哪种 MVC/MVVM 的框架辅助开发; 如果是移动端的网站,需要使用哪种适配模式; 对于前后台数据整合,是用“套页面”的方式还是利用 AJAX 进行前后台数据交互。

### 2.1.4 网站页面策划阶段

当确定基本技术之后,就可以开始将一个成熟的想法转换为实际内容了。

在网站页面策划阶段当中,与客户做沟通的通常是产品经理,客户提出需求,产品经理需要理解需求,之后,通过与客户协商,明确并理解客户的每个需求和各种要求,书写出网站功能描述书,制作出能够传达出需求的“原型图”(RP 图)。就网站的设计风格、网站模块内容达成共识,并建立需求变更制度与流程,方便后期的制作与完善。

之后,产品经理将功能描述书和 RP 图提交给技术总监或技术主管(当然也有可能是开发人员,具体拿给谁取决于公司的开发团队规模以及团队架构)。保证技术方参与项目开发的所有成员,能够准确、清晰地了解需求,并在开发过程中按照客户的需求去做,而不是为技术而迁就需求。

### 2.1.5 网站设计开发阶段

经过上述阶段之后,工作开始移交给开发人员。这个阶段以设计师、前端工程师与后台工程师为主导。

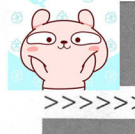
在该阶段中,设计师或 UI(User Interface, 用户界面)设计人员将根据产品经理提供的原型图进行适当分析,然后设计一套前端开发人员能够使用的设计图(PSD 图)。

后台开发工程师根据产品经理提供的原型图,进行后台数据库的构建、数据表字段的确认与前端 API 的设置等工作。

前端开发工程师的工作是在设计师设计完成 PSD 图之后进行的,前端工程师根据设计师的 PSD 图,进行切图工作,并使用具体的 HTML、CSS、JavaScript 代码实现网站的还原。

前端工程师和后台工程师要定期地对接工作,将前端书写的“静态”网站,修改为拥有动态数据的动态网站,也就是用户可以使用的真正的网站。





### 2.1.6 网站测试改进阶段

网站测试并不是在开发彻底完成之后才进行的,通常在开发以及整合过程中,就会进行网站测试的工作,在开发工作结束之后,会进行集中性的测试。

前端主要需要进行网站展示效果和功能的测试,要保证和设计图的一致性,保证在不同浏览器下页面布局与样式的统一、功能的完整。后台方面需要进行网站性能等方面的测试。在前端与后台代码整合之后,前端还需要测试在图片大小或文字数量超出原 PSD 图限制范围时,会不会对页面效果造成影响等。

在网站“试上线”之后,并不意味着这个网站的结束。

从技术层面来说,会发现一些之前测试时没有发现的问题,此时要继续进行网站的修改;还会发现有些功能的设置上并不是很完善,于是进行网站版本的迭代与更新(可以简单地理解为功能的增加、删除和修改)。

从市场层面来说,当前的网站并不为人所知,定期的维护与更新能够提升网站在搜索引擎中的自然排名(关于 SEO 方面的知识在第 5 章中会详细讲解),市场方面的推广与营销策略,则是通过“资金投入”的方式提升网站的“知名度”,为网站增加用户量等。

### 2.1.7 前端工程师负责的部分

在整个流程中,出现了客户、产品经理、技术经理、前端、后台、设计、测试、推广运维等各类职位。每个职位、团队中的每个人,都是网站得以实现的必不可少的组成部分,也是这些人的工作内容,共同组成了整个开发流程。

对于前端开发工程师来说,主要工作在网站设计开发阶段以及网站测试改进阶段,主要进行代码的开发、整合以及上线、调试、修改等工作。

## 2.2 第一个 HTML 文件

### 2.2.1 创建基本的网站文件夹

#### 1. 推荐的文件夹结构

一个网站中包含各式各样的文件,例如,网页 HTML 文件、图像文件、样式文件、脚本文件等,那么如何放置这些文件呢?推荐的文件夹结构如图 2.2 所示。

**备注:**在此给出的文件夹是一个相对比较完整的文件夹结构,可根据项目的不同,选用其中不同的内容,常规网站当中通常都会包含存放网页文件的文件夹,以及存放图像、JS 脚本、CSS 样式表的文件夹。

#### 2. cn 与 en

根据网站类型的不同,cn 和 en 文件夹也有所不同。对于中英文双语的网站,需要提供 cn 和 en 文件夹,cn 文件夹用于存放中文站的 HTML 文件,而 en 文件夹用于存放英



图 2.2 文件夹层次



网站的 HTML 文件；如果只存在一门语言(大部分网站如此)，只需要保留一个 cn 文件夹即可，当然此时 cn 文件夹也可以更换名称，如 html 文件夹等。

### 3. 关于首页的存放位置

网站的首页(index.html 文件)可以根据具体情况而定，可以与 cn(或 html)、en 等文件夹同级别，也可以放置于 cn 或 en 文件夹当中，与其他的 HTML 文件放置在一起。无论如何操作，均须遵循一个原则：分类清晰且易于操作。

## 2.2.2 创建第一个 HTML 文件

在 Sublime 中新建一个文件，存储为 HTML5demo.html，如图 2.3 所示。

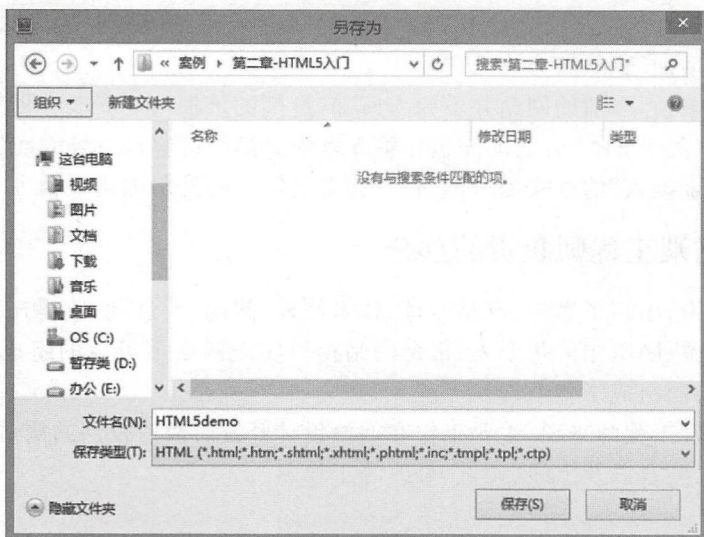


图 2.3 保存文件时的后缀名设置

HTML5demo 是一个文件名，而后的.html 是文件名的后缀，表示文件的类型。然后在编辑器当中输入第一个 HTML 文件的代码。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>我的 HTML5</title>
  <meta name = "keywords" content = "HTML5, 创建文件">
  <meta name = "description" content = "«HTML5 布局之路»我的第一个 HTML5 文件">
</head>
<body>
  <div>这是我的第一个 HTML5 文件</div>
</body>
</html>
```

到目前为止，已创建了第一个 HTML 文件，在编辑器中存储之后，单击鼠标右键，选择





Open in Browser 或“在浏览器中查看”命令,就可以看到网页效果。如果 Sublime 编辑器没有这个功能,可以找到这个文件,然后使用浏览器打开。

网页显示效果如图 2.4 所示。

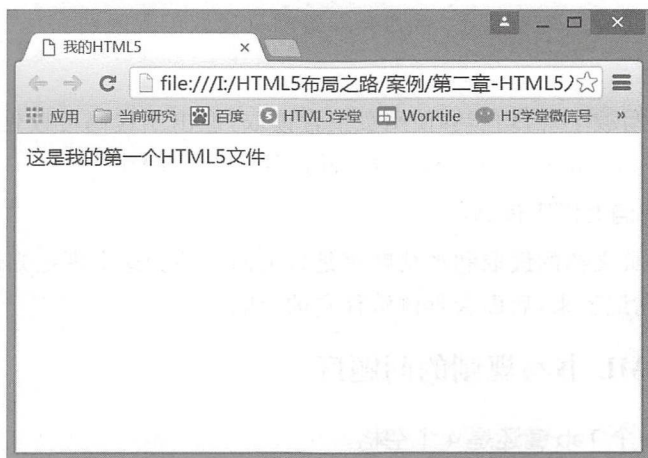


图 2.4 我的第一个 HTML5 文件的预览图

### 2.2.3 HTML 标签的书写规则

#### 1. HTML 文件的后缀名需要设置为 html

在 HTML 文件中书写结构代码(即 HTML 的各类标签),因此需要为文件设置 .html 的后缀。

#### 2. HTML 的各类标签通常成对出现

HTML 的各类标签,大部分都是成对出现,有开有关,如<html></html>。其中,html 是标签的名称,<html>代表 html 这个标签的开始,</html>代表 html 这个标签的结束。

在 HTML 的所有标签当中,也有几个特殊的标签,它们并不会成对出现,这些标签包括 meta、link、img、hr、br、input、col。它们有两种书写方式,一种方法是只书写开标签,另一种方法是书写一个开标签,在标签的>之前,添加/符号,表示标签的结束。

代码实例:

```
<meta charset = "UTF - 8">  
<meta charset = "UTF - 8" />
```

备注: 常用标签会在 2.3.7 节进行详细介绍。

#### 3. 合理缩进

可使用键盘上的 Tab 键进行缩进,每一个级别在当前状态下缩进一次。所谓级别,指的是包含关系,比如 meta 包含在 head 中,此时 meta 就要在 head 标签的缩进基础上再缩进一次。

例外情况: 对于<html></html>标签,还包含着<head></head>和<body></body>标签,由于这两个标签内部的元素才是具体网页中的元素,因此,此处可以对 head 和 body



标签进行缩进,也可以不进行缩进。本书推荐不进行缩进。

#### 4. 英文标点

在 HTML 文件中,非内容部分的所有符号,均为英文半角状态下的标点符号。此处不能够使用中文标点或英文全角状态,否则会导致网页读取出现问题。

#### 5. 标签名小写

标签名使用大写或小写,均能够使网页正常读取,但出于代码可读性考虑,通常要求标签名小写。即采用`<html></html>`,而非`<HTML></HTML>`。

#### 6. 网页文档的自上而下读取

浏览器针对网页文档的读取和加载顺序是自上而下的。这个理论知识会在后面很多地方使用到,因此请先记下来,后面会具体解释它的影响。

### 2.2.4 HTML 书写规则的问题区

#### 1. 缩进选用一个 Tab 键还是 4 个空格

业内有过缩进选用 Tab 键还是空格的争论。在缩进方面,Tab 键和空格这两种方法的特点并不相同。

##### 1) Tab 键与空格缩进的特点

Tab 键:在书写代码时会很方便,但是在不同编辑器当中,距离大小有可能不同。在编辑器当中,一个 Tab 键的长度大小存在一个默认值,这个默认值可以是两个空格长度,也可以是 4 个,还可以是 8 个。由于团队开发当中,每个人的编辑器默认情况下可能不一致,从而有可能造成 A 的代码放到 B 的编辑器中时,缩进的长度发生变化。

4 个空格:在每个编辑器当中,空格大小表现一致,其不足之处在于,代码书写时会比较麻烦,需要按多次空格键。

##### 2) 如何选择缩进方式

其实并不需要纠结这个问题,各个编辑器都提供了“空格”和 Tab 键的相互转换。因此,完全可以使用 Tab 键进行书写,在书写完毕之后,将 Tab 键转换为空格,之后生成最终文件即可。当然,有些公司并不要求使用空格,那开发工程师就更方便了。

最终文件采用哪种方式,需要根据公司要求而定。如果公司没有硬性要求,那么自己的代码一定要保持一致性,不要出现“10 个文件中,3 个使用的是空格进行缩进,7 个使用的是 Tab 键进行缩进”这种现象。

如图 2.5 所示,在 Sublime 编辑器中,可以进行选项卡宽度的设定,也可以实现空格与 Tab 键的互相转换。

#### 2. 是不是只有 .html 才能够被浏览器读取

首先,.html 文件能够被浏览器读取是毋庸置疑的,但并非只有 .html 文件才能够被浏览器读取。

当前端制作完成页面之后,与后台进行代码整合,.html 文件中就被加入了 PHP、Java 或 ASP 等各类后台代码,静态网页也被转换为拥有即时性数据的动态网页,此时为了让这个文件支持动态数据,会将文件的类型进行修改,即将文件的后缀名从原来的 html 改成



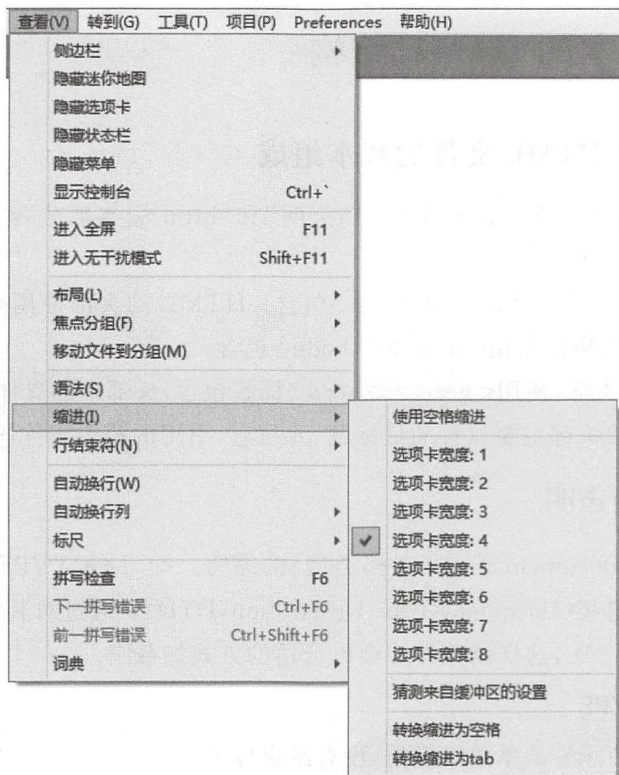


图 2.5 空格与缩进的互相转换

php、jsp、asp、htm 等。对于这些文件，浏览器也是可以正常读取的。

### 3. 一个网站的网页如何命名

1) 首页固定命名为 index.html 或 default.html

.html 为后缀，代码前后台整合后，可以更换为其他后缀。

要求首页固定命名的原因在于：浏览器在访问到一个域名时，首先查看的是当前目录下有没有名称为 index 或 default 的文件，如果有则自动读取，如果没有，则给出的是当前目录下的文件夹结构菜单。

2) 其他网页命名不要使用中文或汉语拼音

网页命名使用小写英文字母，不需要使用大写，对于网页命名来说，通常要求具有语义性，从而便于解读。比如：根据具体的页面内容，转换为英文来命名。

3) 可通过前缀区分不同类别的网页

对于列表页类型的网页名称，可以在前面添加 list 前缀；对于内容页类型的网页名称，可以在前面添加 arc 前缀。list 和 arc 分别表示列表和文章。通过这种方式让自己的网页名称更加清晰。

4) 遵循公司标准

除了上述的基本要求之外，很多成型的互联网公司，都还有自己一些“附加”的命名规则，可按照公司的标准进行网页的命名。



## 2.3 基本的 HTML 结构

### 2.3.1 一个 HTML 文件的基本组成

一个基本的 HTML 文件需要由“文档声明”和“html 标签部分”组成。“html 标签部分”包含头部和内容区两部分。

“html 标签部分”使用< html ></html>包含；HTML 的头部使用< head ></head>标签包含；HTML 的内容区使用< body ></body>包含。

HTML 文件的头部,使用< head ></head>标签包含,头部中的具体内容并不会展示在网页内容中。在文件头部主要放置的是标题、元信息、引用的相关文件等。

### 2.3.2 文档声明

DOCTYPE 是 Document Type(文档类型)的缩写。<! DOCTYPE >元素用于声明一个页面的文档类型定义(Document Type Declaration,DTD)。通过对其定义,告知浏览器当前文件的类型是 HTML,这样浏览器才会以合适的方式加载它。

#### 1. 关于 DOCTYPE

- (1) DOCTYPE 标签是单独出现的,没有结束标签;
- (2) 文档类型定义在 HTML 文档的第一行,在 html 标签之前;
- (3) 文档类型,会使浏览器使用相应标准加载网页并显示;
- (4) 文档不定义 DOCTYPE,浏览器将无法获知 HTML 或 XHTML 文档的类型,有些浏览器会进入怪异解析模式;
- (5) DOCTYPE 与 doctype 同理,不区分大小写,均可使用。

#### 2. 文档声明的具体种类

HTML 从发明到现在已经将近三十年的历史,从 1.0 版本到如今的 5.1 版本,有 HTML1.0、HTML2.0、HTML3.2(已废除)、HTML4.0、HTML4.01、HTML5、XHTML1.0、XHTML2.0(草案)、DHTML 等各类标准,在 HTML5.0 推出之前,HTML4.01 存在三种不同类型的文档声明。

#### 3. 当前使用的文档声明

众多的文档声明当中,当前使用的是 HTML5.0 的版本,即文档声明的代码书写为:

```
<!DOCTYPE HTML>(即<!doctype html>)
```

#### 4. 扩展知识——HTML4.01 的文档声明种类

HTML4.01 文档声明的类型主要有如下几种:过渡定义类型、严格定义类型、框架定义类型。

- (1) 过渡定义类型: HTML4.01 文档过渡定义类型,此类型定义的文档,对于标记和属



性的语法要求并不是很严格,可以使用 HTML 中的标签与元素包括一些修饰性标签(如 u、b 等标签),不可以使用框架。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

(2) 严格定义类型: HTML4.01 文档严格定义类型,此类型定义的文档,对于文档内的代码要求比较严格,不允许使用任何表现层的标记和属性(如 u、b 等标签),不可以使用框架。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

(3) 框架定义类型: HTML4.01 文档框架定义类型,除 frameset 元素取代了 body 元素之外,此类型等同于 HTML4.01 文档过渡定义类型,但可以使用框架。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

### 2.3.3 title 标题

title 表示的是一个网页的标题,这个标题并不显示在网页的界面中,而是显示在打开的浏览器的选项卡上。

### 2.3.4 meta 元信息

#### 1. 元信息

所谓元信息,指的是对信息进行描述的信息。

网页是一个信息,那么这个网页有什么属性呢?比如:这个网页主要讲解的是谁?作者是谁?采用了什么语言?这一系列“信息”都是在对这个网页进行描述,而网页又是信息,所以为了区分开这两种“信息”,就把 meta 称为“元信息”。

meta 是用来在 HTML 文档中模拟 HTTP 的响应头报文。meta 标签位于网页的 <head>与</head>中,meta 的属性有两种: name 和 http-equiv。

name 属性主要用于描述网页,对应于 content 属性(网页内容),以便于搜索引擎机器人查找、分类。

http-equiv 属性,相当于是 http 的文件头作用,它可以向浏览器传递一些有用的信息,以帮助正确和精确地显示网页内容,与之对应的 content 属性,其内容就是各个参数的变量值。

#### 2. 元信息基本语法

基本语法:

```
<meta 属性 = "该属性下的子属性" content = "具体子属性对应的属性值">
```

代码实例：

```
<meta name = "author" content = "lili">
<meta http-equiv = "refresh" content = "4;url = http://www.h5course.com">
```

### 3. name 属性

name 属性下包含如下几种常见的子属性：generator、keywords、description、author、copyright、robots。对于一些不太常见的子属性，在此不再介绍。

generator：代码的生成工具，用于向搜索引擎说明网页是使用哪种工具软件开发的。

keywords：关键词，用于向搜索引擎说明这个网页的关键词是什么，通常会填写多个词语，不同词语直接使用英文的逗号分隔开（也可以使用其他标点符号作为分隔符）。

description：描述信息，用于向搜索引擎概括性地介绍这个网页的主要内容是什么。

author：作者，用于向搜索引擎说明网站的作者。

copyright：版权，用于向搜索引擎说明版权信息。

renderer：渲染内核，用于控制浏览器以什么内核进行渲染。

robots：机器人，用于向搜索引擎说明文件的检索方式，即哪些页面需要索引，哪些页面不需要索引，主要包括 all、none、index、noindex、follow 和 nofollow 几种属性值。

(1) 设定为 all：文件将被检索，且页面上的链接可以被查询。

(2) 设定为 none：文件将不被检索，且页面上的链接不可以被查询。

(3) 设定为 index：文件将被检索。

(4) 设定为 follow：页面上的链接可以被查询。

(5) 设定为 noindex：文件将不被检索，但页面上的链接可以被查询（不让 robot/spider 登录）。

(6) 设定为 nofollow：文件将不被检索，页面上的链接可以被查询（不让 robot/spider 顺着此页链接向下深入地探查）。

代码实例：

```
<meta name = "generator" content = "Sublime Text">
<meta name = "keywords" content = "HTML5, 布局, 前端, Web">
<meta name = "description" content = "HTML5 布局之路, 是一本从开发实战角度出发, 按照开发流程讲解 HTML5 的书籍.">
<meta name = "author" content = "刘国利 - 独行冰海">
<meta name = "copyright" content = "刘国利">
<meta name = "robots" content = "all">
```

### 4. http-equiv 属性

http-equiv 属性下包含如下几种常见的子属性：content-type、content-language、refresh、expires、pragma、set-cookie、pics-label、windows-target、page-enter、page-exit。

content-type：用于设置网页内容的编码类型。

content-language：用于说明网页所使用的语言。

refresh：用于定时让网页在指定时间内刷新或跳转到其他页面。



expires: 用于设定网页的到期时间,一旦过期则必须到服务器上重新调用。需要注意的是必须使用 GMT 时间格式。

pragma: 用于设定禁止浏览器从本地机的缓存中调阅页面内容,设定后一旦离开网页就无法从 Cache 中再调出。

set-cookie: 用于设置 cookie(缓存)的过期时间,如果网页过期,存盘的 cookie 将被删除。需要注意的也是必须使用 GMT 时间格式。

pics-label: 用于进行网页等级评定,在 IE 的 Internet 选项中有一项内容设置,可以防止浏览一些受限制的网站。

windows-target: 用于告知网页在当前窗口中以独立页面显示,可以防止自己的网页被别人当作一个 frame 页调用,即防止被钓鱼。

page-enter: 用于设定进入网页时的特殊效果,注意当前页面不能够是一个 frame 页面。

page-exit: 用于设定离开网页时的特殊效果,注意当前页面不能够是一个 frame 页面。

代码实例:

```
<meta http-equiv="Content-Type" content="text/html";charset="gb_2312-80">
<meta http-equiv="Content-Language" content="zh-CN">
<meta http-equiv="Refresh" content="n;url=http://yourlink">
<meta http-equiv="Expires" content="Sun,31 July 2016 00:20:00 GMT">
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="set-cookie" content="Mon,12 May 2001 00:20:00 GMT">
<meta http-equiv="Pics-label" content="";>
<meta http-equiv="windows-Target" content="_top">
<meta http-equiv="Page-Enter" content="revealTrans(duration=10,transtion=50)">
<meta http-equiv="Page-Exit" content="revealTrans(duration=20,transtion=6)">
```

## 5. 目前使用频繁的元信息

当前使用频繁的元信息设置主要包括字符编码、关键字、描述信息、自动刷新或跳转、独立页面显示。

其中,字符编码与网页中文字是否乱码相关;关键字、描述信息,与网站在搜索引擎中的搜索排名相关;自动刷新或跳转用于实现一些特殊功能需求(如 404 页面中隔 5s 后跳转回主页的功能);独立页面显示,主要是防止自己的网站被钓鱼。

## 6. 设置字符编码

代码实例:

```
<meta charset="UTF-8">
```

代码解析:

这句话用来声明整个网页的字符串集的格式。UTF-8 是一种编码格式,为世界通用,如果不设为 UTF-8,那么在不支持其他字符集格式的网页中,所有的汉字都会变为乱码。UTF 也可以使用小写的 utf。

#### 代码书写位置：

该行代码的位置，建议书写在< head></head>标签中的第一位置（即作为 head 元素的第一个子元素），放置位置要前于< title></title>。这样放置的主要原因在于：网页文档是自上而下加载的，在 title 当中有时也会使用汉字，如果将 title 标签放置在< meta charset="UTF-8">之前，那么 title 中的中文内容有可能变成乱码，同时在 IE8 等浏览器中，整个网页的加载都会被“阻塞”。

### 7. 自动刷新或跳转页面

#### 代码实例：

```
<meta http-equiv="refresh" content="4;url=http://www.h5course.com">
```

#### 代码解析：

在 4s 之后，页面进行刷新，会自动跳转到 http://www.h5course.com 这个网址。如果在代码当中没有进行 url 的设置，页面仅刷新，而不会跳转。

### 8. 浏览器内核控制

#### 代码实例：

```
<meta name="renderer" content="webkit">
```

#### 代码解析：

代码表示让浏览器以 WebKit 内核进行网页的渲染。

renderer 用于控制浏览器以什么内核进行渲染。在国内的主流浏览器当中（360 浏览器等），大都采用的是双核浏览器，分别是基于谷歌的 WebKit 内核以及基于 IE 的 Trident 内核。通常情况下，会优先选用 WebKit 内核进行网站渲染，对于很少的一些网站，才会使用 IE 内核渲染以保证页面的兼容问题。

可以通过 meta 的设置，告知浏览器该网页应当采用哪种类型进行渲染，浏览器在读取到这个 meta 设置之后，会切换相应的内核，并且将这个行为应用于这个二级域名下的所有网址（简单地理解为该页面所处文件夹下的所有子页面）。当前 360 安全浏览器已经完全实现了这种技术。

renderer 的取值有三种，为 webkit、ie-comp、ie-stand，分别表示极速模式（WebKit 内核）、IE 兼容内核（ie-comp）、IE 标准内核（ie-stand）。区分大小写，因此请注意书写格式。

如果需要对浏览器以某种内核渲染，设置相应的 renderer 即可。相对使用较多的是 WebKit 内核设置。

## 2.3.5 HTML 文件的内容区

< body></body>所包含的部分就是 HTML 文件的内容区，这个区域中的具体内容，默认情况下都会展示在网页当中。前端开发工程师就是通过在这里书写标签以及文字，让浏览网页的每个用户查看到相应信息和内容。

## 2.3.6 HTML 注释

在网站开发中，有时出于可读性的考虑，会为 HTML 代码添加注释（标注出某些标签



是哪个模块),代码可读性高会更便于开发工程师本人以及团队成员阅读;在调试代码时,若发现有些布局错乱,为了快速排查出错误也会使用 HTML 注释(将一部分结构代码注释掉,然后查看布局情况,如果布局恢复正常,说明问题出现在注释掉的一段代码当中;如果布局依旧错乱,则说明问题出现在未注释的代码当中。通过几次的注释和调试,能够将问题快速定位)。

如图 2.6 所示,在 HTML 当中,使用<!-- 注释内容-->来表示注释内容。其中,<!--表示注释的开始,-->表示注释的结束,该注释可以注释单行内容,也可以注释多行内容。

当浏览器读取到第一个<!--时即注释开始,寻找到第一个-->则注释结束,因此,如果在原有注释外再进行注释,建议先将原有注释删除,防止出现问题。

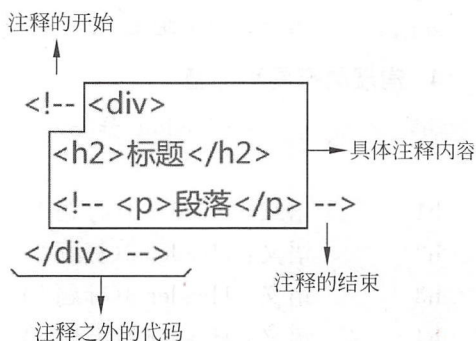


图 2.6 HTML 注释图解

### 2.3.7 网站开发常用标签

#### 1. 具体分类以及讲解顺序

标签具体分类、作用和讲解章节如表 2.1 所示。

表 2.1 标签具体分类、作用和讲解章节

标 签 分 类	标 签 作 用	涉 及 章 节
网页结构类	创建 HTML 文件	第 2 章
文件引入类	为标签添加样式	第 3 章
常规布局类——div	完成网页的整体布局	第 3 章、第 4 章
常规布局类——其他	选择合理的标签实现模块	第 5 章
段落文本类	实现模块内部某一行的布局	第 5 章
特殊功能类	提升模块的可用性	第 6 章、第 9 章
表格类	制作表格	第 10 章
表单类	制作表单	第 11 章
其他出现较少的标签	很少使用	不做讲解

#### 2. 网页结构

html	语义：网页文档,用于包含一个 HTML 文件。
body	语义：网页主体/内容区。
head	语义：网页头部。
meta	语义：网页元信息。
title	语义：网页标题。

### 3. CSS 与 JS 引入类

link	语义：网页外部链接，将外部文件(如 CSS)引入到当前文件当中。
style	语义：网页样式。
script	语义：网页脚本，用于定义行为代码，或将行为代码引入到当前文件当中。

### 4. 常规类布局用标签

div	语义：Division(分隔)。	在文档中定义一块区域，即包含框。
h1	语义：Header 1(标题 1)。	定义一级标题。
h2	语义：Header 2(标题 2)。	定义二级标题。
h3	语义：Header 3(标题 3)。	定义三级标题。
h4	语义：Header 4(标题 4)。	定义四级标题。
h5	语义：Header 5(标题 5)。	定义五级标题。
h6	语义：Header 6(标题 6)。	定义六级标题。
p	语义：Paragraph(段落)。	定义段落结构。
ol	语义：Ordered List(有序列表)。	定义具有一定顺序的排序列表。
ul	语义：Unordered List(无序列表)。	定义没有一定顺序的列表。
li	语义：List Item(列表项目)。	定义每一条具体列表项。
dl	语义：Definition List(自定义列表)。	以自定义的方式进行列表。
dt	语义：Definition Term(定义术语)。	定义自定义列表中的标题。
dd	语义：Definition Description(定义描述)。	定义自定义列表的内容。
hr	语义：Horizontal Rule(水平线)。	定义一条分隔线。

### 5. 特殊功能用标签

a	语义：Anchor(定义锚)。	超链接。
img	语义：Image(定义图像)。	定义图像包含框。

### 6. 段落文本处理用标签

span	语义：Span(范围)。	在文本行中定义一个区域，即包含框。
em	语义：Emphasized Text(加重文本)。	定义文本为重要。
strong	语义：Strong Text(加重文本)。	定义文本为很重要。
br	语义：Break(换行)。	定义文本内部换行(注：不是创建新段落)。
b	语义：Bold Text(加粗文本)。	定义文本加粗。
i	语义：Italic Text(斜体文本)。	定义文本倾斜。

### 7. 表格类标签

table	语义：Table(表)。	定义一个表格。
caption	语义：Table Caption(表格标题)。	定义一个表格的标题。
thead	语义：Table Header(表格头部)。	定义一个表格的头部区域。



tbody	语义: Table Body(表格主体)。	定义一个表格的主体(内容)区域。
tfoot	语义: Table Footer(表格脚)。	定义一个表格的脚部区域。
col	语义: Table Columns(表格列)。	定义一个表格的列区域。
colgroup	语义: Groups of Table Columns (数据列组)。	定义一个表格的数据列组。
tr	语义: Table Row(行)。	定义一个表格行。
td	语义: Table Data Cell(表格数据单元)。	定义一个表格单元格。
th	语义: Table Header Cell(表头数据单元)。	定义一个表格的表头单元格(列标题)。

## 8. 表单类标签

form	语义: Form(形状)。	定义表单。
fieldset	语义: Field Set(域组)。	定义表单的字段域。
legend	语义: Legend(图例)。	定义表单字段域的标题。
label	语义: Label(标签)。	定义表单的控制标签。
input	语义: Input Field(文本区域)。	定义表单输入域。
textarea	语义: Text Area(文本区域)。	定义表单属于区域。
select	语义: Selectable List(可选择的列表)。	定义下拉菜单或列表框。
option	语义: Option(选项)。	定义下拉选项或列表选项。
optgroup	语义: Option Group(选项组)。	定义下拉选项组。
button	语义: Push Button(发送按钮)。	定义表单的发送按钮。

## 9. 其他出现较少的标签

address	语义: Address(地址)。	定义地址。
abbr	语义: Abbreviation(缩写词)。	定义缩写词。
acronym	语义: Acronym(取首字母的缩写词)。	定义取首字母的缩写词。
big	语义: Big Text(大文本)。	定义文本增大。
blockquote	语义: Block Quotation(区块引用语)。	定义大块内容的引用。
cite	语义: Citation(引用)。	定义引文。
code	语义: Code Text(源代码)。	定义计算机源代码。
dfn	语义: Defines(定义条目)。	定义条目。
kbd	语义: Keyboard Text(键盘文本)。	定义键盘键。
q	语义: Quotation(引用语)。	定义一段文字中的引用短语。
small	语义: Small Text(小文本)。	定义文本缩小。
sub	语义: Subscripted Text(下标文本)。	定义文本下标。
sup	语义: Superscripted Text(上标文本)。	定义文本上标。
samp	语义: Sample(示例)。	定义样本示例。
tt	语义: Teletype Text(打印机文本)。	定义打印机字体。
u	语义: Underlined Text(下画线文本)。	定义文本下画线。
var	语义: Variable(变量)。	定义变量。

### 2.3.8 基本 HTML 结构的问题区

(1) 网页的描述信息和关键词并不会显示在内容区,可以不书写吗?

建议进行书写,虽然网页 head 中的 meta 信息并不会显示在网页当中,但是当网页被搜索引擎收录时,显示在搜索结果中的就是这个网页的描述信息,如图 2.7 所示。



图 2.7 网页的描述信息在搜索引擎中的显示效果

(2) 字符编码有哪些?

常见的字符编码主要有 ASCII、Unicode、GBK、GB 2312 等。

世界上存在着多种编码方式,同一个二进制数字可以被解释成不同的符号。因此,要想打开一个文本文件,就必须知道它的编码方式,否则用错误的编码方式解读,就会出现乱码。有时电子邮件会出现乱码的问题,这就是因为发信人和收信人使用的编码方式不一样所导致的。

① ASCII 编码用来表示英文。

② GBK 和 GB2312 表示简体中文(GB2312 编码大约包含六千多个汉字,GBK 编码是对 GB2312 编码的扩充,容纳的汉字更多)。

③ Unicode 编码包含世界上所有的字符,并且每一个符号都是独一无二的。其编码的标准有很多种,比如: UTF-8, UTF-16, UTF-32 等。注意: UTF-8 是 Unicode 字符的实现方式之一。

(3) 什么时候会产生乱码现象?

HTML 文件的内容有一个字符编码问题,HTML 文件本身又有一个字符编码问题,浏览器在解读时也有一个字符编码问题。

第一种编码通过 < meta > 标签进行设置,第二种编码是在存储文件时进行设置(Sublime 默认是存储为 UTF-8 的编码格式,txt 文本文档会默认存储为 ANSI 的编码格式),第三种编码,在默认情况下为 UTF-8,也可以通过菜单进行修改(如图 2.8 所示)。



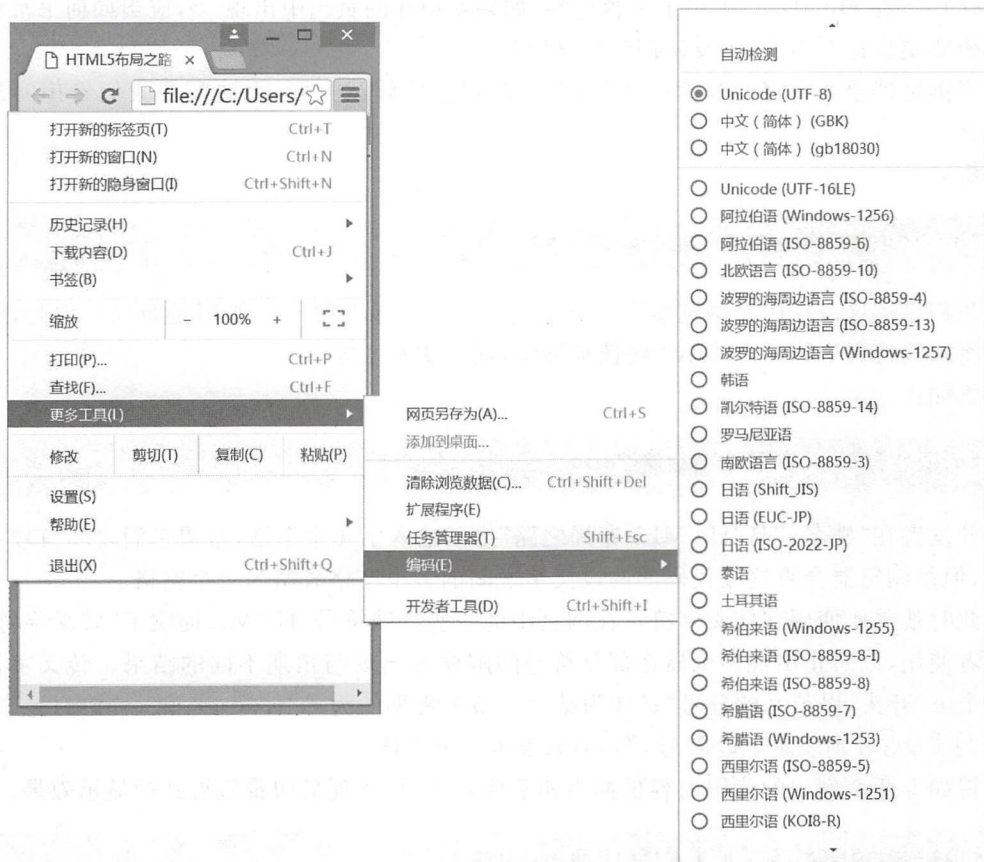


图 2.8 浏览器编码设置

如果不希望网页中的中文文本显示为乱码,这三种编码格式必须统一!否则,都有可能造成中文乱码现象。

(4) `< meta charset="UTF-8"/>`与`< meta charset="UTF-8">`哪个是正确的?

两种书写方法都是正确的。

对于单标签来说,可以使用`/>`来表示标签的结束,也可以不书写`/`。

(5) 包含属性值的引号,使用单引号还是双引号?

`< meta charset="UTF-8">`与`< meta charset='UTF-8'>`这两段代码,只是在引号的地方有所不同。在 HTML5 的开发当中,对于属性值外部的引号,到底有没有明确规定呢?

① 系统默认为双引号,使用 Sublime 编辑器开发时,会使用一些快捷键生成一些标签,标签中自带属性所使用的引号均为双引号。

② 单引号也符合语法要求,使用单引号的 HTML 文件能够正常解析,并不会引发问题。

③ 自我应当保持一致,也就是说,在一个 HTML 文件当中,所有同类型的引号要么都是双引号,要么都是单引号,不要出现“一会儿双引号一会儿单引号”的现象。

④ 在此建议使用双引号,主要是由于系统为双引,开发者又要自我保持一致,使用双引会相对比较方便。

想要实现这个需求,需要使用到转义字符。

当浏览器遵循一定的解读规律时,在一些符号的解读方面,无法实现我们希望实现的效果。

例如：

<div><div>标签的主要作用是布局</div>

我们希望这段代码在浏览器当中显示为: `<div>` 标签的主要作用是布局。但是, 浏览器会将这段代码中第二个 `<div>` 解读成标签, 而非文本内容。

再如：

<div>姓名: HTML5 布局之路</div>

开发者在“姓名:”与“HTML5 布局之路”之间输入了 4 个空格,希望它们之间有一定的间隔,但是浏览器会将这段代码的两段文字之间的多个空格省略为一个空格。

此时就需要使用到转义字符。在网页中的一些特殊符号,HTML 提供了“转义字符”供开发者使用,以防止出现浏览器在部分符号的解读时出现与预期不同的结果。转义字符均以 `&` 开头,以英文的分号“`;`”作为结束。最为常见的为 `&lt;`、`&gt;` 和 `&nbsp;` 这三种。`&lt;` 与 `&gt;` 分别表示 `<` 和 `>` 符号, `&nbsp;` 表示一个空格。

将如上两个例子的代码内容更换为如下代码,即可实现最初希望得到的显示效果。

[illegible]



## 第3章

# 整体布局(上)——标签尺寸处理



### 3.1 整体布局与整体布局中使用的标签

#### 3.1.1 整体布局

了解基本规则,做好前期开发准备之后,就要真正进入“代码书写”阶段了。网页整体布局,是整个网站具体代码实现时的第一步,该步骤是针对网页进行分析,划分为多个模块,将一个网页解读成几个组成部分,并针对这些组成部分进行位置、大小的控制。第3章和第4章所涉及的具体知识如图3.1所示。



图 3.1 第3章和第4章涉及的知识



在整体布局的过程中,首先会接触并使用到< body >当中的第一种 HTML 元素——div。这种元素也是在整体布局当中会使用到的唯一的 HTML 元素。

然后要涉及关于 CSS 布局方面的一些知识,让一个个 div 能够按照开发工程师的需求和想法,呈现出相应的大小,出现在网页中适当的位置上,如图 3.2 所示。

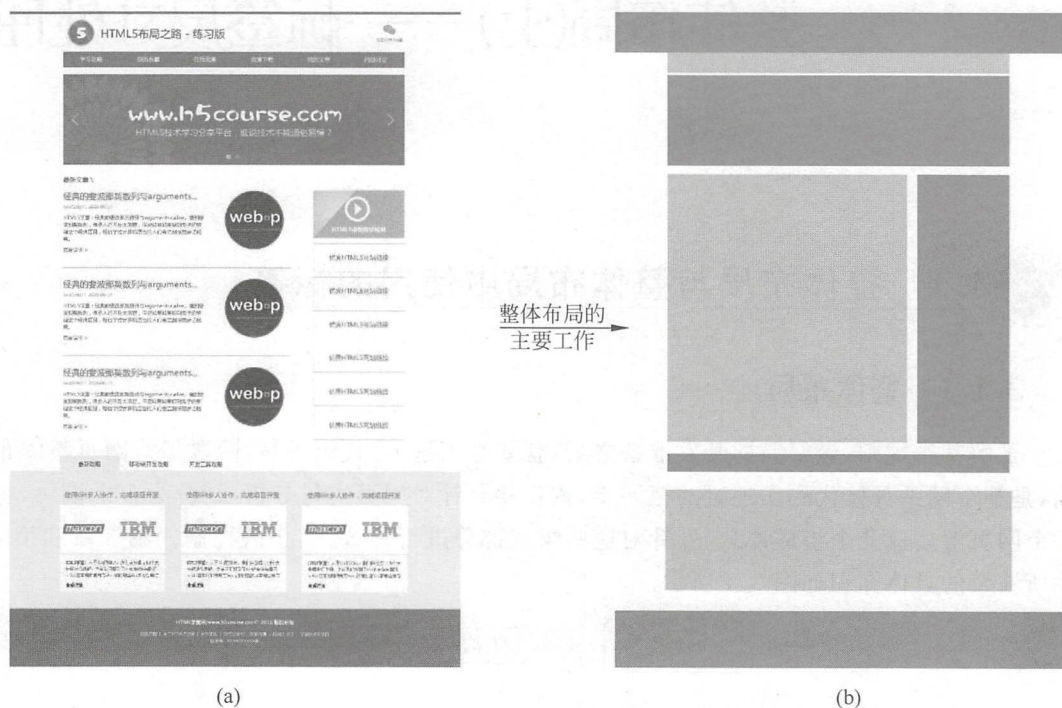


图 3.2 整体布局的主要工作

注:图 3.2(a)为网页的 PSD 设计图,图 3.2(b)为整体布局完成后,HTML 文件在浏览器中的显示效果。

## 3.1.2 div 元素

标签含义:

div 元素没有任何具体含义,在网站开发当中,主要用于网页的布局工作。通过一个的 div,将网站划分为不同的部分,之后再使用其他元素针对每个具体部分进行开发。

div 元素是一个块元素,默认情况下占据父级元素的宽度,由内容撑开高度;可以设置宽度和高度,设置后,依旧独自占据父元素的一行。

注意:关于块元素、行元素的概念,会在第 5 章中详细讲解,此处请先掌握 div 的呈现特点。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - div 元素实例</title>
  <style>
```





```

/* 可以通过如下代码查看到每个 div 的边界 */
div {
    border: 5px solid black;      /* 5 像素黑色实线边框 */
}
</style>
</head>
<body>
    <div>div 元素中的内容,该内容会显示在浏览器中</div>
    <div>
        <div>嵌套在父级 div 中的 div 元素 1</div>
        <div>嵌套在父级 div 中的 div 元素 2</div>
    </div>
</body>
</html>

```



## 3.2 什么是 CSS

55

### 3.2.1 没有 CSS 时代的网页

HTML 比 CSS 早几年诞生,在最初还没有 CSS 的年代,网页的样式显示如图 3.3 所示。

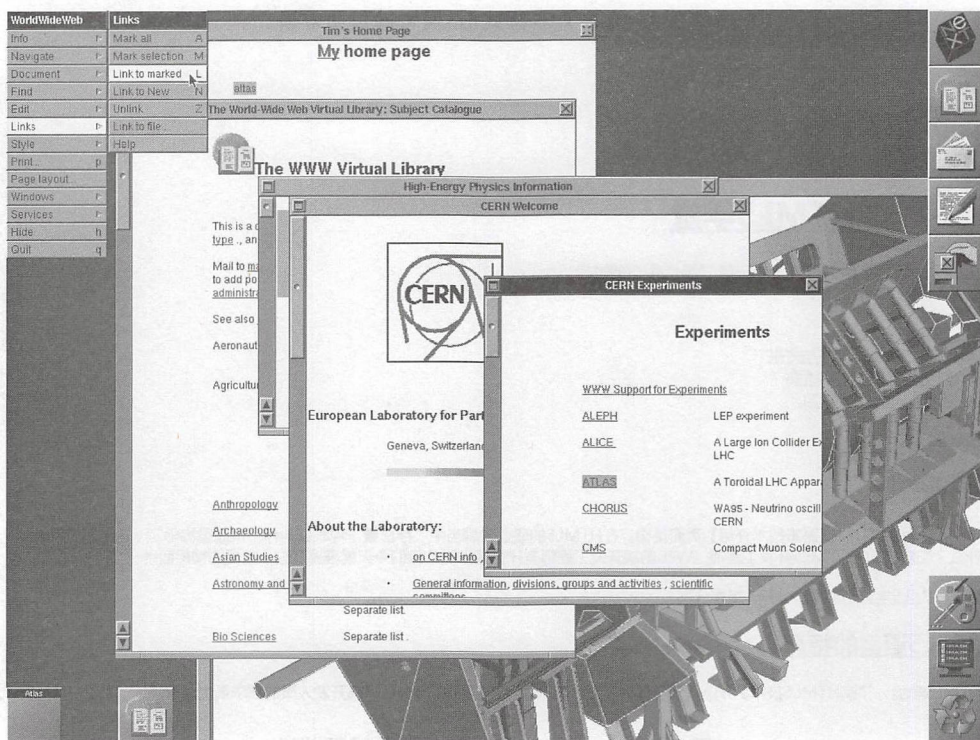


图 3.3 没有 CSS 时代的网页



在当前可以浏览的网站当中,都存在着 CSS 样式代码,可以通过谷歌等浏览器的控制台,去除网页中的 CSS 代码,来查看没有 CSS 时代网页的显示效果。其中,图 3.4 为存在 CSS 样式代码时,HTML5 学堂的页面效果,图 3.5 为去掉 CSS 样式代码之后,HTML5 学堂的页面效果。



图 3.4 存在 CSS 样式时,网页的页面效果

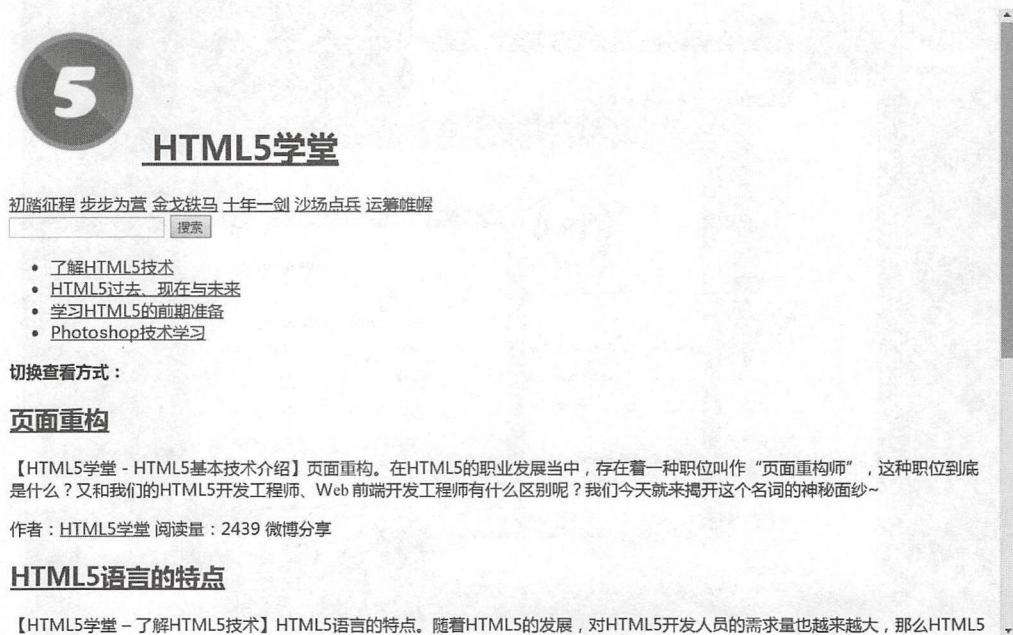


图 3.5 去掉 CSS 样式后,网页的页面效果





### 3.2.2 什么是 CSS

CSS(Cascading Style Sheets),可译为层叠样式表或级联样式表,是一组格式设置规则,主要用于控制 Web 页面的外观。通过使用 CSS 样式设置页面的风格,可将页面的内容与表现形式分离。



## 3.3 CSS 引入方式

### 3.3.1 行内书写——最简单的样式操作方法

#### 1. 最简单样式的操作方法

一个网页中存在结构标签,也存在具体的样式,只有“某些结构标签”应用“某种具体样式”时样式才能够生效,那么如何实现这个关联过程呢?

将 CSS 样式与 HTML 结构“关联”起来的方法有多种,一种最为原始也是最容易理解,看上去似乎是最为简单的书写方式,是直接将“样式”作为标签的“属性”(style)来书写。

基本语法:

```
<标签名 style="具体样式属性: 属性值">内容</标签名>
```

代码实例:

```
<div style="border: 1px solid red">1</div>
```

代码解析:

style 是 div 标签的属性,表示的是“样式”。

style 中的 border 指的是边框这种样式,而“1px solid red”表示的是边框样式为 1 像素、实线、红色。

**备注:**关于边框部分,在后面的盒模型部分当中会做详细讲解,因此先不要急,当前只需要了解标签名、style 属性以及具体的属性名、属性值的关系即可。

#### 2. 实际案例解析

下面是一个实际案例:

在网页(HTML 文件)中存在 10 个 div 标签,前 5 个 div 元素添加“1 像素红色实线边框”的样式,后 5 个 div 添加“1 像素蓝色实线边框”的样式。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 行内书写 CSS</title>
</head>
<body>
```



```

<div style="border: 1px solid red">1</div>
<div style="border: 1px solid red">2</div>
<div style="border: 1px solid red">3</div>
<div style="border: 1px solid red">4</div>
<div style="border: 1px solid red">5</div>
<div style="border: 1px solid blue">6</div>
<div style="border: 1px solid blue">7</div>
<div style="border: 1px solid blue">8</div>
<div style="border: 1px solid blue">9</div>
<div style="border: 1px solid blue">10</div>
</body>
</html>

```

### 3. 客户需求带来的麻烦事

将制作好的网页提交给客户,客户查看网页效果时,感觉红色边框并不好看,提出了修改的要求:“能不能将所有的红色边框全部改为绿色?”于是,修改代码如下。

```

<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 行内书写 CSS</title>
</head>
<body>
  <div style="border: 1px solid green">1</div>
  <div style="border: 1px solid green">2</div>
  <div style="border: 1px solid green">3</div>
  <div style="border: 1px solid green">4</div>
  <div style="border: 1px solid green">5</div>
  <div style="border: 1px solid blue">6</div>
  <div style="border: 1px solid blue">7</div>
  <div style="border: 1px solid blue">8</div>
  <div style="border: 1px solid blue">9</div>
  <div style="border: 1px solid blue">10</div>
</body>
</html>

```

将修改后的网页再次提交给客户,客户觉得:“绿色的边框的确比较好看,能不能将所有的蓝色边框也调整为绿色?”

于是,继续调整代码:

```

<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 行内书写 CSS</title>
</head>
<body>
  <div style="border: 1px solid green">1</div>
  <div style="border: 1px solid green">2</div>

```





```

<div style="border: 1px solid green"> 3 </div>
<div style="border: 1px solid green"> 4 </div>
<div style="border: 1px solid green"> 5 </div>
<div style="border: 1px solid green"> 6 </div>
<div style="border: 1px solid green"> 7 </div>
<div style="border: 1px solid green"> 8 </div>
<div style="border: 1px solid green"> 9 </div>
<div style="border: 1px solid green"> 10 </div>
</body>
</html>

```

当再次修改代码之后,客户查看着绿色的边框说:“或许把边框都变成黑色更好一些。”再一次修改代码:

```

<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 行内书写 CSS</title>
</head>
<body>
  <div style="border: 1px solid black"> 1 </div>
  <div style="border: 1px solid black"> 2 </div>
  <div style="border: 1px solid black"> 3 </div>
  <div style="border: 1px solid black"> 4 </div>
  <div style="border: 1px solid black"> 5 </div>
  <div style="border: 1px solid black"> 6 </div>
  <div style="border: 1px solid black"> 7 </div>
  <div style="border: 1px solid black"> 8 </div>
  <div style="border: 1px solid black"> 9 </div>
  <div style="border: 1px solid black"> 10 </div>
</body>
</html>

```

客户看过修改后的代码说:“看了这么多种颜色,我还是觉得没有边框最好。”最后,之前书写、修改的样式代码都被删掉了,之前的工作似乎是在消磨时间,作为开发工程师这样的遭遇又何止一次……

### 3.3.2 内部书写——简化样式操作

#### 1. 简化样式操作的“内部书写”

经过几次需求更改,作为开发工程师的你,有没有感觉到“style 属性”的书写方式产生了大量重复代码?一旦涉及修改,操作起来也异常麻烦?

发明 CSS 的人与我们有着类似的感受,于是就提供了另外两种方法,来解决如此麻烦的问题。

代码实例:

```

<!doctype html>
<html>

```



```
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 内部书写 CSS 样式</title>
  <style>
    div {
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</body>
</html>
```

在这段代码中,将 CSS 的属性、属性值放在了一个< style></style>标签当中,这种书写方法叫作 CSS 样式的“内部书写”。

#### 代码解析:

< style></style>标签当中,div 表示选择器,这个选择器能够找到的标签就是 body 元素中的所有 div 元素;大括号{}包含具体的样式,这些样式将应用于选择器选择到的所有标签;“border: 1px solid black;”表示的是具体样式代码。

### 2. “内部书写”方法的原理与优势

原理:将样式代码统一放置于 style 标签当中,然后通过“选择器”,将规定的样式和相应的标签建立联系。

优势:使用“内部书写”的方法来操作样式时更加方便快捷。在进行 CSS 样式修改时,并不需要翻阅 HTML 代码,也不需要去针对每个 HTML 标签处理样式,只需要调整 style 标签中的样式,就可以针对所有相应标签进行样式的修改。

### 3. 你可能在思考的问题

可能你会考虑:这种改变让所有的 div 都发生样式变化,但是最初的需求是一半红色一半蓝色,这种书写方式无法满足之前的需求。

如果能够想到这个问题,说明你更深入地进行了思考。不过,先把这个问题放一放,本章稍后会讲解各种各样的选择器,通过各种选择器,可以很灵活地选择想要的标签,自然能够轻松地解决这个问题了。

## 3.3.3 外部引入——控制多页面样式

### 1. 内部书写在网站中的不足

掌握在 HTML 文件中书写 style 标签进行相应元素的样式控制之后,还要面对这样的





场景：一个网站当中拥有 10 个甚至更多的 HTML 页面，在这 10 个 HTML 页面中，必然会拥有相同的部分，例如，网页的头部、底部、导航区、某些模块等。对于开发工程师来说，如果希望能够一次性控制这些不同 HTML 文件中相同模块的样式，内部书写的方法就无法实现这个需求了。

## 2. link 标签

link 标签的作用：将外部样式表文件链接到 HTML 文档中。

link 较完整写法：<link rel="stylesheet" type="text/css" href="css/index.css">

(1) href：定义外部样式表(CSS)文件的地址，该属性值可以使用 URL 地址的格式进行设置，可以使用相对路径或绝对路径。

(2) type：定义链接文档的类型。type="text/css"表示当前标签包含内容的文本类型为 CSS 样式代码。

(3) rel、rev：定义链接外部文件的关联类型，也就是设置文件默认应用的是哪种类型的链接文件。rel="stylesheet"表示浏览器在默认状态下应该先执行样式表的文件类型。

(4) 除此之外，link 还包含 title、media 等属性，由于较少在开发中使用，在此不再讲解。

## 3. 字符编码

外部引入的 CSS 文件，需要进行字符编码的设置：

```
@charset"UTF-8";
```

@charset 命令用于定义外部引入的 CSS 文件的字符编码，该命令需要在 CSS 文件中的最前面定义，且在当前的 CSS 文件当中只允许出现一次。如果开发工程师没有为 CSS 文件定义编码格式，这个文件将按照被引入到的 HTML 文件的编码，来确定自己的编码。

字符编码的代码要放在 CSS 文件的最前端，由于网页代码的加载顺序自上而下，放在最前面能够让后面所有的代码都按照这种字符编码格式进行编码和解析。

## 4. 路径

### 1) 什么是路径

从字面上可以将路径理解为通向某个目标的道路。

如果需要在 HTML 文件当中引入一个 CSS 文件，那么从当前的 HTML 文件出发，怎样才能找到这个 CSS 文件呢，是找当前文件级别下的某个文件，还是找当前位置的父级的某个文件，或者找当前位置的子级的某个文件？

以当前 HTML 文件为起点，以要引入的文件为终点，构成的道路就是“路径”。

### 2) 何处需要使用路径

在前端开发中，除了在 HTML 中引入 CSS、JS 文件时会使用路径之外，还会在 HTML 或 CSS 文件引入图片、字体、多媒体文件等情况下用到路径。应该说，只要是需要引入的文件，都需要使用到路径。

### 3) 路径的种类

路径有两类，一种是相对路径，一种是绝对路径。

### 4) 相对路径

相对路径：以引用文件的网页所在位置为参考基础，而建立出的目录路径。当保存于



不同目录的网页文件引用同一个文件时,所使用的路径将不相同,故称之为相对。例如:

```
../images/h5course.jpg
```

5) 绝对路径

绝对路径: 以 Web 站点根目录为参考基础的目录路径。之所以称为绝对,意指当所有网页引用同一个文件时,所使用的路径都是一样的。例如:

```
G:/2016/h5course/images/h5course.jpg
```

6) 开发时的选用

对于绝对路径和相对路径的选择,在开发中通常使用相对路径。这主要是因为当开发工程师在客户端书写好代码,并上传到服务端时,服务器端的盘符、具体的文件位置与客户端并不相同,如果使用绝对路径,会在路径上出现错误。

7) 路径的特殊符号

- ../表示当前文件所在层级的上一层级。
- ./表示当前文件所在的层级。
- /表示根目录。

8) 相对路径书写实例

图 3.6 展示了一个相对路径书写实例。  
当前的文件夹结构中存在 cn 文件夹、css 文件夹以及 index.html 文件。在 cn 文件夹中有一个 list.html 文件,css 文件夹当中有两个 CSS 文件 index.css 和 reset.css。

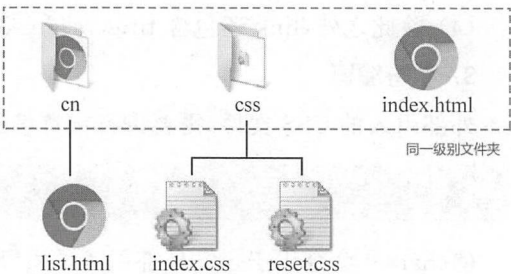


图 3.6 相对路径书写实例

代码实例 1: index.html 中引用 reset.css 文件: 首先找到同级的 css 文件夹,之后找到这个文件夹中的 reset.css 文件。

```
css/reset.css
```

代码实例 2: list.html 中引用 reset.css 文件: 首先找到 list.html 文件的父级(cn 文件夹),然后找到 cn 文件夹的同级 css 文件夹,再从 css 文件夹当中找到 reset.css 文件。

```
../css/reset.css
```

5. 外部引入的书写方法

以上面案例中的 10 个 div 为例,假设存在多个 HTML 文件,这 10 个 div 在这些 HTML 文件中属于公共模块,即多个 HTML 文件当中存在这 10 个 div 元素,且在这些文件中的 10 个 div 元素样式相同。

一旦客户需求发生变更,开发工程师就需要打开所有包含“公共模块”的 HTML 文件,依次进行编辑和修改,这时的操作变得异常复杂和麻烦。第三种样式引入方式——“外部引入”能够很好地解决这个问题。



外部引入的原理：外部引入是将 CSS 代码书写在一个 CSS 文件当中，然后通过 link 标签，使用正确的路径，将这个 CSS 文件引入到 HTML 文件当中。在 CSS 文件中的具体代码遵循了内部书写的基本规则，也涉及选择器、具体样式等。

代码实例：HTML 文件中的具体内容如下。

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 外部引入 CSS</title>
  <link rel = "stylesheet" href = "css/index.css">
</head>
<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</body>
</html>
```

代码实例：CSS 文件中的具体内容如下。

```
@charset 'UTF - 8';
div {
  border: 1px solid black;
}
```

代码注意事项：

将 HTML 文件命名为 index.html，CSS 文件名称为 index.css（index 为文件的名称，.html、.css 均表示文件的后缀名）。

CSS 文件处于 HTML 文件所在文件夹中的 css 文件夹内。即某文件夹包含 index.html 和 css 文件夹，而 css 文件夹包含 index.css。

文件夹名与文件名的大小写不要写错，此处 link 标签中的 href 表示路径，如果文件位置放置错误或文件名写错，CSS 文件并不能够被引入到 HTML 文件中，样式不会生效。

外部引入这种方式将 HTML 结构与 CSS 样式完全分离成了两个文件，实现了结构与样式的分离，增强了网页结构的扩展性，提高了代码的可维护性。在多个 HTML 文件当中都引入了同一个 CSS 文件时，这个 CSS 文件中的代码修改，就能够引发这些 HTML 文件中相应部分的变化，代码的操作与维护就变得更加快捷与轻松了。

### 3.3.4 CSS 三种常见引入方式比较

页面中使用 CSS 的方式主要有三种，分别是行内书写（行内添加定义 style 属性值）、内

部书写(页面当中使用< style>标签调用样式)和外部引入(通过外部链接,引入 CSS 文件)。

下面比较一下这三种 CSS 的引入方式,看看在实战当中,它们分别应用于什么情况。

1. 控制性

控制性比较如表 3.1 所示。

表 3.1 CSS 三种引入方式——控制性比较

引入方式	行内书写	内部书写	外部引入
控制性	最弱	中等	最强

标签内书写: 只控制当前标签。

头部书写: 通过一个选择器能够控制当前页面中所有符合条件的标签。

外部引入: 一个 CSS 文件能够控制多个页面,实现了结构与样式相分离。

2. 维护性

维护性比较如表 3.2 所示。

表 3.2 CSS 三种引入方式——维护性比较

引入方式	行内书写	内部书写	外部引入
维护性	不便于维护	维护性一般	便于维护

标签内书写: 修改起来最不方便。

头部书写: 对于多个网页中都存在的公共样式,修改起来并不方便,需要多次重复修改。

外部引入: 对于多个网页中都存在的公共样式,只需要修改一次就可以完成所有的修改。

3. 代码量

代码量比较如表 3.3 所示。

表 3.3 CSS 三种引入方式——代码量的比较

引入方式	行内书写	内部书写	外部引入
代码量	存在大量的冗余代码	对于单页面来说,没有冗余代码 对于整站,依旧存在大量冗余代码	对于整个网站来说,不存在冗余代码,代码量较少,有效地利用了浏览器有缓存机制 对于单页面来说,存在冗余代码

标签内书写: 文件大小变大,从而导致加载速度变慢,用户体验变差。

头部书写: 对于单页面来说,所有的 CSS 代码都是有用的,加载速度较快。但是,由于各个网页中存在相似模块,因此对于整站来说,存在冗余代码,代码量较大。

外部引入: 访问站点时,会下载 CSS 文件,当整站 CSS 文件下载完毕之后,再去加载其他页面,速度会非常快(涉及浏览器缓存)。但是,对于单页面来说,特别是第一个页面,浏览器需要下载的 CSS 文件当中,除了包含当前页面的样式之外,也会包含其他页面的 CSS 代码,因此,针对这一个页面来说,就出现了冗余代码,加载速度相对较慢。





4. 服务器请求压力

服务器请求压力的比较如表 3.4 所示。

表 3.4 CSS 三种引入方式——服务器请求压力的比较

引入方式	行内书写	内部书写	外部引入
服务器请求压力	无	无	有可能造成

外部引入：数据(图片、文件)能够呈现在网页中,需要两步——由客户端向服务器发出申请,再由服务器将数据发回。当客户端的用户多次向服务器索要数据,同时服务器还很忙时,就会面临“服务器请求压力”的问题。

5. 应用场景

CSS 三种引入方式的应用场景如表 3.5 所示。

表 3.5 CSS 三种引入方式的应用场景

引入方式	行内书写	内部书写	外部引入
应用场景	特殊情况(基于其优先级特点)	比较大型网站的首页	大部分小型网站

标签内书写：特殊情况指的是“找不到哪里出现问题,但是项目又着急上线,需要立即修改某个部分的样式”。

头部书写：比较大型网站的首页,需要以最快速度加载出首页,把样式代码书写在网页头部必然是最好的选择。

外部引入：小型网站没有太多的页面,也不会生成大量的代码,因此可以将整站的代码放置在同一个文件当中。

6. 综述

在实际的开发当中,通常以外部引入为主,在必要的时候选用内部书写和行内书写的方法。切忌背定义,要根据具体的开发要求灵活变通。

3.3.5 外部引入 CSS 的扩展知识

1. @import 的基本语法

除了介绍到的 link 标签能够外部引入 CSS 文件之外,还可以使用@import。

使用 link 方法引入的代码实例：

```
<link rel = "stylesheet" rev = "stylesheet" href = "CSS 文件" type = "text/css" media = "all" />
```

使用@import 的方法引入的代码实例：

```
<style type = "text/css" media = "screen">
    @import url("CSS 文件");
</style>
```



## 2. link 与 @import 的区别与使用原则

这两种方法虽然都能够实现外部引入 CSS,但是存在一定的区别。也正是由于这些区别,才导致多数人在实际开发中放弃了使用 @import 的引入方式。

(1) link 是 XHTML 标签,除了加载 CSS 外,还可以定义 RSS 等其他事务; @import 属于 CSS 范畴,只能加载 CSS。

(2) link 引用 CSS 时,在页面载入时同时加载; @import 需要页面网页完全载入以后加载。

(3) link 是 XHTML 标签,无兼容问题; @import 是在 CSS2.1 中提出的,低版本的浏览器不支持。

(4) link 支持使用 JavaScript 控制 DOM 去改变样式;而 @import 不支持。

### 3.3.6 CSS 引入方式的问题区

温馨提醒:对于理论类的问题请先思考,实战类的问题可以先动手输代码,得出基本结论,再看答案,学习方法和基本知识同等重要。

(1) 如何理解 `<div style="border: 1px solid red"></div>` 代码中 style 与 div 的关系?

① 代码的基本含义。

div 是一个标签元素,style 是标签的一种属性,而“border: 1px solid red”是 style 这种属性的具体属性值。对于一个标签来说,能够拥有多种属性,另外,具体的属性值可能会再细分。

```
<标签名 标签的某一种属性名字="具体属性值" 标签的某一种属性名字="具体属性值"></标签名>
```

② 借助生活的类似实例加深理解。

可以借助生活中的实际情况来理解这段代码:将把元素理解成人,每个人都有基本特征这种属性,而基本特征当中又可以细分为性别、身高、体重等(当然,每个人也可以有爱好、教育经历、工作经验这些属性等)。于是这段代码就成了这个样子:

```
<人 基本特征属性="性别:男;身高:180cm;体重:80kg;" 教育经历="本科:.....;高中:.....;初中:.....;小学:.....;" 工作经验="....." 个人爱好="....."></人>
```

③ 回看 link 标签。

此时,应该能够更深入地理解外部引入 CSS 时使用的 link 标签了。

```
<link rel="stylesheet" href="../../css/index.css" type="text/css" />
```

本段代码当中,link 是一个标签,拥有 rel、href、type 三种属性,其中,rel 属性的属性值是“stylesheet”; href 属性的属性值是 index.css 这个文件的文件路径,即“../../css/index.css”; type 属性的属性值是“text/css”。

(2) 网页中有冗余代码有何影响?

代码冗余度大,通常是由于在书写代码时,采用了错误的 CSS 引入方式,或者没有将相同模块的样式合并起来进行书写。大量的冗余代码会让工程师的维护工作变得麻烦,也会





增加文件的大小,从而对加载速度造成影响。

#### ① 对于工程师来说,代码变得难于维护。

开发工程师书写完成基本的网页代码,并不是就结束了,后期需要大量的修改和维护。换言之,工程师还要多次面对自己写过的代码,并针对其进行修改。

如果在网页制作过程中,开发工程师采用的是冗余度较高的 CSS 行内书写的方式,一旦进行修改,就不得不查看很多文件,并一一排查,确定是否需要修改。一个很简单的、很小的修改操作,就可能会耗费掉工程师非常多的时间。如果工程师采用< style >标签在 HTML 文件中内部书写 CSS,对于一些修改也会比较繁杂,虽然不用一个个地排查标签,但是还是需要打开各个 HTML 文件,进行一些重复操作,在操作过程中也容易造成文件的遗忘和丢失,出现比较低级的错误。

#### ② 用户体验下降。

冗余的代码,也就意味着文件增大,当文件增大之后,整个网站的加载速度会下降,而用户体验会变差。网站的每一个浏览者,都希望能够快速打开一个网站,而不是等待三五秒之后网站才逐渐呈现在眼前。用户体验变差,很有可能造成用户流失,从而对企业造成致命性的影响。毕竟,对于广大的用户来说,只要一个网站不是唯一的、必须使用的,那么他们完全可以抛弃当前平台,去找另一个功能类似但用户体验更好的网站。

(3) style 标签必须书写在头部吗? 放在其他位置会不会无法实现样式效果?

可以在编辑器当中尝试如下代码:

```
<!doctype html >
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - style 位置的影响</title>
</head>
<body>
  <div>第一个 div</div>
  <style>
    div {
      border: 1px solid red;
    }
  </style>
  <div>第二个 div</div>
</body>
</html>
```

在浏览器显示效果当中,两个 div 均存在 1 像素红色实线边框。也就是说,style 标签书写在 body 标签中时代码生效,div 得到了 style 标签中设置的具体样式效果。

style 并不仅仅可以放置在 body 元素之内,还可以放在某个具体标签之内,也可以和其他标签同级,还可以放置在< html >标签之内,< body ></body >之后,甚至可以放在</html >标签的后面。

虽然 style 标签的位置随意,但是并不推荐将它放置在< head ></head >标签之外。主要原因有以下两点。

#### ① 不便于维护: 当 style 标签放置到 body 标签中,工程师在后期维护时,就需要花额



外的时间进行寻找,在密密麻麻的代码当中寻找某种样式并进行修改,或者调试是否因为某种样式而引发的问题,绝对是一件让人心烦的事情。

② 会引起页面回流与重绘,从而降低加载速度。网页是由 DOM Tree(也就是各个标签)和样式结构体组合后构建成的 render tree。render tree 构建完成后,浏览器就会根据 render tree 来进行页面的绘制。

- 页面回流:当 render tree 中的一部分(或全部)因为元素的规模尺寸、布局、隐藏等改变而引起的页面重新渲染(或者叫作重新构建绘制)。
- 页面重绘:当 render tree 中的一些元素需要更新属性,但这些属性只会影响元素的外观、风格,而不会影响到元素的布局,此类的页面渲染叫作页面重绘。

由于网页文档是自上而下加载并解析的,假设将 style 标签书写在了</body>的后面,在读取到 style 标签之前,整个网页文档已经渲染得差不多了,这时候突然发现存在一个 style 标签,并且 style 标签中的内容针对网页中某些元素的样式进行了设置,于是浏览器只好重新为每个元素计算样式,再加载出来。这会导致网页的加载速度变慢。

(4) 对于引入 CSS 文件的 link 标签,可不可以书写在网页文档的其他位置?

与上面提到的< style></style>类似,link 标签也可以放置在网页文档的其他地方。出于与< style>标签同样的原因,会将< link>标签放置在 head 标签当中,先于 HTML 结构加载。



## 3.4 CSS 选择器

通过引入方式,在 HTML 页面中“放入”了 CSS 样式,接下来要做的操作就是让相应的标签应用定义好的 CSS 样式,这里会涉及“选择器”的基本知识。

### 3.4.1 生活中的“选择器”——找人

今天学校举办了一场“舞会”,邀请所有的同学出席,现在你和你宿舍的好友们已经把要穿的衣服、饰品准备好了,之后就是要“选择”谁“穿着”什么样的“衣服或饰品”出席“舞会”。

例如,我们希望:

- (1) “身份证号码”是 XXX 的人穿着“长裙”;
- (2) “班级”是“HTML5-1 班”的人,手上戴着“腕表”;
- (3) “男性”打“领带”。

将“希望”书写成需求,并进行合理拆解,如表 3.6 所示。

表 3.6 找人的基本需求拆解

需 求	选 择 条 件	服装要求(样式)
“身份证号码”是 XXX 的人穿着“长裙”	身份证号是 XXX	长裙
“HTML5-1 班”的人,手上戴着“腕表”	HTML5-1 班	腕表
“男性”打“领带”	男性	领带

每个人最初都没有“服装要求”(样式),所有的“服装要求”(样式)是根据需求已确定的,而“选择条件”将人和“服装要求”(样式)关联到了一起。





### 3.4.2 CSS 选择器的基本语法

```
选择器名{  
    属性名: 属性值;  
    属性名: 属性值;  
    :  
}
```

### 3.4.3 CSS 基本选择器

在生活的“舞会”当中,通过“选择条件”,让不同的人 and 不同的服装要求建立关系。在代码世界当中,HTML 标签就相当于参加舞会的每个人,而 CSS 就是“服装要求”(样式),连接这个人和礼服的桥梁,就是“选择器”。

CSS 当中存在三种基本选择器,分别被称为:ID 选择器、类名选择器与标签名选择器。也就是说,CSS 为我们提供了三种最基本的方法,来为“标签”与“样式”建立联系。

对于 ID 选择器,可以理解成身份证;类名选择器,可以理解为班级(或一个类);标签名选择器,可以理解为性别。

#### 1. 通过身份证找人——ID 选择器

基本语法:

```
#选择器名{  
    属性名: 属性值;  
    属性名: 属性值;  
    :  
}
```

选择器代码:

```
#con {  
    border: 1px solid red;  
}
```

代码实例:

```
<!doctype html>  
<html>  
<head>  
    <meta charset = "UTF - 8">  
    <title>HTML5 布局之路 - ID 选择器</title>  
    <style>  
        #con {  
            border: 1px solid red;  
        }  
    </style>  
</head>  
<body>
```



```
<div id="con">1</div>
<div>2</div>
<div>3</div>
</body>
</html>
```

#### 代码解析：

#con {border: 1px solid red;}这句代码,表示找到 ID 名为 con 的元素,并为元素设置 1 像素、实线、红色边框。

身份证号码,是生活中标识人们身份的一串数字,那么身份证号码(ID 选择器)有什么特点呢?

(1) 属于我们每个人自己、独一无二,并不会与其他人重复。

(2) 通过身份证来找人,只能找到一个人。

(3) 号码需要定义。身份证号码,并不是在人出生时就规定好的,而是需要到相关部门登记,在登记后我们才拥有了这个号码。在 HTML 代码当中,设置 id 属性,就是定义了 id 名,在 CSS 中通过这个 id 名来寻找相应的元素,此时“#id 名”就是创建了一个 ID 选择器,让样式与 ID 选择器对应的标签建立联系。

## 2. 通过班级找人——类名选择器

#### 基本语法：

```
.选择器名{
    属性名: 属性值;
    属性名: 属性值;
    :
}
```

#### 选择器代码：

```
.con {
    border: 1px solid blue;
}
```

#### 代码实例：

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 类名选择器</title>
    <style>
        .con {
            border: 1px solid blue;
        }
    </style>
</head>
<body>
    <div class="con">1</div>
```





```
<div>2</div>
<div class="con">3</div>
</body>
</html>
```

#### 代码解析:

.con {border: 1px solid blue;} 这句代码,表示找到类名(class)为 con 的元素,并为元素设置 1 像素、实线、蓝色的边框。

班级(即类名选择器)的特点如下。

(1) 有可能选择到的是一个(班里只有一个人),也有可能选择到的是多个(每个班级可以有 multiple 个人,很多标签都可以起同一个类名)。

(2) 班级(类名选择器),并非是天生所有的属性,而是需要后期定义的。在 HTML 代码当中,class 属性的设置,就是在定义类名;而 CSS 中的“.类名”就是创建了一个类名选择器,让其中的样式与类名选择器对应的标签建立联系。

### 3. 通过性别找人——标签名选择器

#### 基本语法:

```
选择器名{
    属性名: 属性值;
    属性名: 属性值;
    :
}
```

#### 选择器代码:

```
div {
    border: 1px solid blue;
}
```

#### 代码实例:

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 标签名选择器</title>
    <style>
        div {
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <div>1</div>
    <div>2</div>
    <div>3</div>
</body>
</html>
```



代码解析：

div{border: 1px solid black;}这句代码,表示找到标签名为 div 的元素,并为元素设置 1 像素、实线、黑色的边框。

性别(即标签名选择器)的特点如下。

(1) 如果说找到所有的男性,那么全世界的男性都符合这个条件,可见,这种方式找人,找到的人数有可能是一个,也有可能是多个。

(2) 每个人在出生的时候就已经自带了性别属性,并不是说别人把自己定义成什么性别,就是什么性别,因此,对于性别(标签名选择器)这个特点,并不需要进行定义。在代码当中:标签名选择器,使用的就是标签,标签名就如同性别,只不过标签的类别比较多,不只有男、女,此处可以理解成动物的种类(如灵长类、两栖类)。此时,直接将“标签名”作为选择器名,就能够让样式与相应的标签建立联系。

(3) 注意: HTML 当中不存在的标签,是不能够使用标签名选择器来进行书写的。

4. 三种选择器的特点总结

三种基本选择器的比较如表 3.7 所示,三种基本选择器的用法如表 3.8 所示。

表 3.7 三种基本选择器的比较

选择器类型	生活中的同类物	选择范围	精准度	是否需要定义
ID 选择器	身份证号	最小	高	是
类选择器	班级号	较大	一般	是
标签名选择器	性别	最大	最差	否

表 3.8 三种基本选择器的用法

选择器类型	标签中的操作	CSS 中的使用方法
ID 选择器	定义标签的 id 属性和属性值	# 选择器名{ 属性:属性值; }
类选择器	定义标签的 class 属性和属性值	. 选择器名{ 属性:属性值; }
标签名选择器	无	标签名{ 属性:属性值; }

3.4.4 样式冲突的问题

一个人,可以同时属于“女性”、“1 班”、“身份证号 XXX”,一个标签也可以被多种选择器同时选中,此时就会出现样式冲突的问题。在了解具体的样式冲突之前,先看几个代码实例。

样式冲突问题——代码实例 1:

```
<!doctype html>  
<html>
```



```
< head>
  < meta charset = "UTF - 8">
  < title>HTML5 布局之路 - 样式冲突问题 1</title>
  < style>
    div {
      border: 1px solid red;
    }
    .con {
      border: 1px solid black;
    }
  </style>
</head>
< body>
  < div> 1</div>
  < div class = "con"> 2</div>
  < div> 3</div>
</body>
</html>
```

### 样式冲突问题——代码实例 2:

```
<!doctype html>
< html>
< head>
  < meta charset = "UTF - 8">
  < title>HTML5 布局之路 - 样式冲突问题 2</title>
  < style>
    div {
      border: 1px solid red;
    }
    div {
      border: 1px solid black;
    }
  </style>
</head>
< body>
  < div> 1</div>
  < div> 2</div>
  < div> 3</div>
</body>
</html>
```

### 样式冲突问题——代码实例 3:

```
<!doctype html>
< html>
< head>
  < meta charset = "UTF - 8">
  < title>HTML5 布局之路 - 样式冲突问题 3</title>
  < style>
    div {
      width: 100px;
      height: 100px;
      border: 1px solid black;
```

```
        }
        #test {
            background: #39f;
        }
        .con {
            border: 10px solid black;
        }
    </style>
</head>
<body>
    <div>1</div>
    <div id="test" class="con">2</div>
    <div>3</div>
</body>
</html>
```

在如上的三个例子当中,虽然都是样式冲突,但是也各有不同。有些选择器中书写的是同样的样式;有些选择器书写的是不同的样式;有些标签能够被多个选择器选择到,而且在不同的选择器当中设置了不同的样式。

那么,在这些样式中,哪些样式才是标签最终真正展示出来的样式呢?

此时,需要先了解一下“选择器的优先级”的基本知识。

3.4.5 生活中的“优先级”——谁是老大

“优先级”其实可以理解成“谁说了算”,如表 3.9 所示。

表 3.9 三种基本选择器的身份设定

ID 选择器	类名选择器	标签名选择
老师	班长	组长

经历了一周的学习,到了周五上午,组长通知说:“明天需要过来上课”,而班长也发出了通知:“明天放假,不需要过来上课”,那么此时,你会听谁的呢?

相信你会在心里比较组长和班长这两个人哪个人的级别高,谁说话算数,就听谁的。

3.4.6 CSS 选择器优先级

1. 行内样式与 CSS 选择器的优先级

在行内,使用 style 属性书写的样式,优先级最高(但是通常都不这么写样式)。

ID 选择器的优先级其次,类名选择器优先级再次,优先级最弱的是标签名选择器,如表 3.10 所示。

表 3.10 style 属性以及三种基本选择器的优先级

作为标签属性 style,设置的各类样式	ID 选择器优先级	类名选择器优先级	标签名选择器优先级
1000	0100	0010	0001



## 2. 不同的选择器的优先级问题——对应样式冲突的第一个例子

实例解析,如图 3.7 所示。

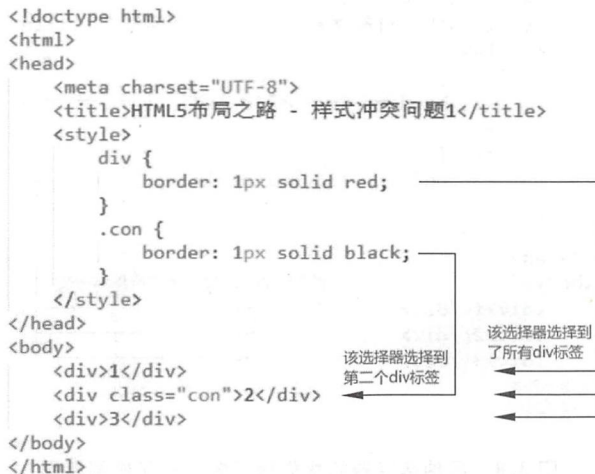


图 3.7 不同选择器的优先级问题——实例解析图

< style>标签中的第一个选择器“div”是标签名选择器,第二个选择器“. con”是类名选择器。

第一个选择器能够选择到 body 中的所有 div 元素,也就是为每个 div 元素均设置了 1 像素实线的红色边框。

第二个选择器选择到的是类名为 con 的元素,也就是上面代码中的第二个 div 元素。为其设置了 1 像素实线的黑色边框。

那么此时,第二个 div 元素应该是什么样式呢? 边框是黑色的还是红色的呢?

上面提到,标签名选择器的优先级是 0001,而类名选择器的优先级是 0010。这两个选择器相比,很明显,0010 要大于 0001,也就意味着,“. con”这个选择器的优先级高于“div”这个选择器,因此显示出来的是黑色边框。

## 3. 同种选择器的优先级问题——对应样式冲突的第二个例子

依旧使用之前的身份设定。周五上午,班长通知大家:“明天需要过来上课”,可是到了周五的下午,班长又发出了通知:“明天放假,不需要过来上课了。”那么此时,你是遵循班长的第一个通知还是第二个通知呢?

相信你会听后面的“消息”。在 CSS 当中,由于网页的读取是自上而下的,因此,对于同种优先级的选择器,后书写的代码生效。

实例解析如图 3.8 所示。

< style>标签中有两个选择器,都是“div”这种标签名选择器,优先级均为 0001,这两个选择器均能够选择到 body 中的所有 div 元素。

第一个选择器为所有的 div 设置了 1 像素的红色实线边框。第二个选择器为所有的 div 设置了 1 像素的黑色实线边框。这两个选择器优先级相同,但是,由于第二个选择器在后面,浏览器就会用“第二个选择器对应的边框样式”覆盖掉“第一个选择器对应的边框样

```

<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5布局之路 - 样式冲突问题2</title>
  <style>
    div {
      border: 1px solid red;
    }
    div {
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
</body>
</html>

```

图 3.8 同种选择器的优先级问题——实例解析图

式”。最终三个 div 显示出来的，均为黑色边框。

#### 4. 样式并不是只有覆盖——对应样式冲突的第三个例子

当不同的选择器都选择到了某个标签，并且针对这个标签设置了相同的样式，这些样式会被彼此覆盖掉。但是，当不同的选择器都选择到了某个标签，而不同选择器设置的样式各不相同，会发生什么？

生活中的实例：

组长(标签名选择器)发布通知：“明天放假”。

班长(类名选择器)发布通知：“明天上课；明天需要携带学生证”。

老师(ID 选择器)发布通知：“明天需要携带身份证”。

综合之后，消息会变成“明天上课，需要携带学生证、身份证”。

在这些消息当中，有冲突的消息发生了覆盖(明天放假还是上课)，而没有冲突的消息(携带学生证、携带身份证)并没有产生任何的覆盖，而是进行了叠加。

对于 CSS 样式，针对一个元素的同一种属性而设置的样式，会根据选择器的优先级进行覆盖；如果不同的选择器针对元素的不同属性进行了样式设置，这个元素会同时拥有所有符合条件的属性样式。

实例解析如图 3.9 所示。

“div”这个选择器，会选择到所有的 div 元素；“#test”这个选择器会选择到 id 名为 test 的元素；“.con”这个选择器会选择到 class(类)名为 con 的元素。

对于 body 中的第二个 div 元素，均符合这三种选择器的选择条件。那么这时，这个 div 会是什么样式呢？

同时拥有这三种选择器中的样式，但是“border”的样式发生了重复，此时需要比较哪个选择器的优先级更高。



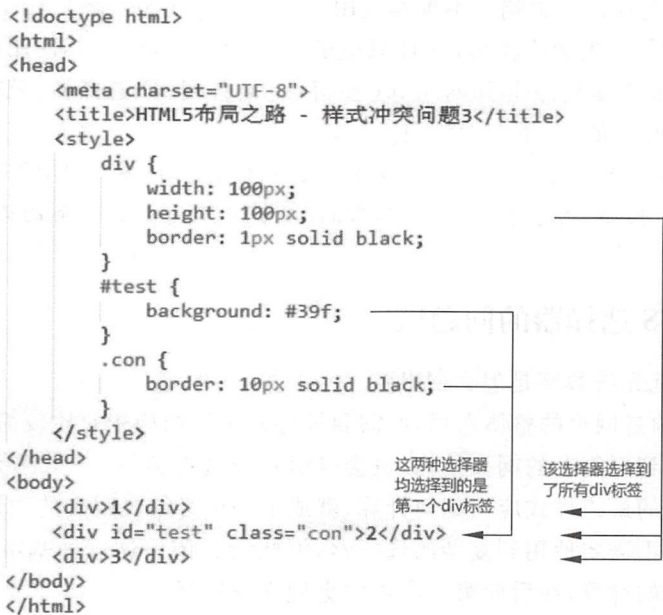


图 3.9 样式并不是只有覆盖——实例解析图

当前第二个 div 元素的样式为：

```
width: 100px;
height: 100px;
border: 10px solid black;
background: #39f;
```

3.4.7 行内的 style 属性

除了上述几种基本选择器外，如前所述可以使用行内的 style 属性进行样式的设置。行内 style 属性中定义的样式，优先级要比所有的选择器都高，在实际开发当中，由于这种方式没有实现样式和结构的相分离，并不推荐使用，在此也不再详细讲解。

基本语法：

```
<标签名 style="属性名：属性值；属性名：属性值；">具体内容</标签名>
```

3.4.8 选择器的使用原则

在网站实战当中，如何合理地选用“三种基本选择器”呢？

- (1) 在此强烈推荐新手使用“类名选择器”。之所以选择“类名选择器”，主要原因在于：类名选择器可以同时选择多个元素，类似的元素可以使用同一个类名，比 ID 选择器操作起来更灵活；由于类名选择器必须定义，只有定义类名的标签才能够应用样式，并不会对网页中其他元素造成不必要的影响。
- (2) 标签名选择器，由于其选择范围过广，会导致网页当中所有的同种类型标签都会被

选中,此时很容易造成样式影响。不推荐使用。

(3) ID 选择器,优先级比较高,并且只能够选择到一个,适合其使用的“环境”并不多。另外,ID 选择器,通常是给原生 JavaScript 预留,一旦看到 ID 选择器,就能够想到“在这里应该是有 JS 功能存在的”。不推荐使用。

注意:可能你会觉得这三种选择器的操作并不足够灵活,不要紧,CSS 还存在更多功能更强的“选择器”。但是,由于当前的“整体布局”功能并不需要太复杂的选择器,因此在第 5 章中再详细讲解。

### 3.4.9 CSS 选择器的问题区

(1) 0010 的优先级数字是怎么回事?

当前在学习的是网页的整体布局,所需要处理的布局结构相对比较简单,后期的一些模块布局当中,会遇到更复杂的网页结构,也会讲解各类选择器,当一个标签符合多个选择器的选择条件时,它的最终样式应当如何计算,就成了一个必须要解决的问题。由于当前处于整体布局阶段,并不需要使用到复杂的选择器,因此请先记住这三种基本选择器的优先级,对于 0010 所涉及的计算,在后面第 5 章当中会做详细解释。

(2) 开发当中是不是不能使用 id 选择器和标签名选择器?

当然不是!

样式重置方面,会使用到标签名选择器(样式重置会在 3.6 节当中讲到)。

在网页的其他地方,id 选择器和标签名选择器也是能够使用的。

在前面“选择器选用”时,之所以提醒大家不要随意使用,主要是由于这两种选择器很容易出现样式覆盖,导致标签的最终显示效果出现问题。

选择器种类的具体选用,要根据网页的实际情况而定。例如,如果整个网站当中,只存在一处表格,需要使用到表格类元素,那么可以使用标签名选择器,因为它并不会对其他任何地方造成影响。但是,即便如此,依旧不推荐使用标签名选择器(除样式重置外)。在开发时,需要考虑后期的维护和调整,以刚才表格需求为例,一旦后期网站发生调整,在网站中添加了四处表格,就需要调整原有的样式代码,维护工作就会变得很不方便。



## 3.5 CSS 编码规范

同样都能够实现代码,为何还要考虑编码规范?

(1) 从维护角度来说,编码在绝大多数情况下是拿给人来看的,只是偶尔让机器读一下。了解基本的编码规范,对自己日后的维护,或者自己小伙伴的维护是有推进作用的。

(2) 优秀的代码会让团队工作事半功倍,不同的编码习惯,其实也直接影响着薪资。

### 3.5.1 CSS 注释

CSS 代码当中,使用 `/**` 来表示注释,在 `/*` 与 `*/` 之间书写具体的注释内容,被注释的内容并不会被浏览器解读,因此可以使用中文注释。在书写代码时,有时需要花几个工作日来完成某一个模块或页面;有时一些 CSS 代码只是测试使用、后期需要删除;有时书写的一些 CSS 代码只是出于某种考虑添加到原有代码当中的。合理的注释能够让开发者更



清晰某段代码的意义,大幅提升代码的可读性。

CSS 注释,可以针对单行文本进行注释,也可以针对多行文本进行注释。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - CSS 代码注释</title>
  <style>
    div {
      width: 100px;           /* 此处书写的是代码注释 */
      height: 100px;         /* 一段 CSS 代码注释,
      可以是单行的,也可以是多行的,
      并不会受到换行等问题的影响 */
      border: 1px solid black;
      /* 在书写时,建议将注释内容与符号之间添加一个空格,方便查看 */
    }
  </style>
</head>
<body>
  <div>/* CSS 代码注释在 HTML 代码中不生效 */</div>
</body>
</html>
```

需要注意的是,当浏览器读取到第一个/\*时即为注释开始,之后寻找到第一个\*/即为注释结束,因此,如果在原有注释外再进行注释,建议先将原有注释删除,防止出现问题。CSS 注释如图 3.10 所示。

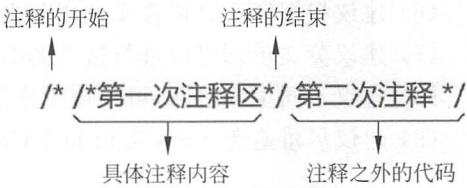


图 3.10 CSS 注释

3.5.2 书写风格

- (1) 每一条规则的大括号{前后加空格。
- (2) 每一条规则结束的大括号}前加空格。
- (3) 属性名冒号之前不加空格,冒号之后加空格。
- (4) 每一个属性值后必须添加分号;单行模式下,分号后加空格。

1. 单行形式

```
div { width: 100px; height: 200px; }
```

2. 多行形式

```
div {
  width: 200px;
  font-size: 16px;
}
```

**注意:**

(1) 空格部分的添加与否,并不会影响到代码的读取。合理地书写空格,能够提升代码的可读性。

(2) 大括号、冒号、分号,则必须使用英文半角状态下的标点符号,并且不能缺失,否则会影响代码读取,页面有可能无法渲染成功。

(3) 在{}中,最后一组的“属性:属性值;”中结尾的分号可以省略,但是并不推荐省略,特别是对于新手来说,省略之后再添加某些样式时,可能忘记添加分号,很容易出现代码问题。

(4) 单行形式和多行形式相比,更加推荐使用多行形式,在阅读方面会更快捷清晰。有些人可能考虑到单行形式的代码量会小一些,的确如此。但是,在实际开发当中,工程师开发时使用的 CSS 文件,需要经过压缩之后,才上传到服务器端,压缩的过程中就有这种“多行形式”到“单行形式”的自动转换。

### 3.5.3 关于类名命名

(1) 不建议使用中文进行命名;

(2) 不建议使用汉语拼音进行命名;

(3) 不建议使用 box1、2、3、4、…、10 等英文+数字的形式进行命名;

(4) 不建议使用过长的英文单词进行命名;

(5) 建议使用英文单词进行类名的命名;

(6) 建议根据模块具体含义选用英文单词,类名要有具体含义;

(7) 建议英文单词可以进行适当的缩写;

(8) 建议不同单词之间,可以使用连字符或下画线进行分隔,如: .list-tit、.list\_tit;

(9) 建议尽量避免 class 与 id 重名(id 通常是 JavaScript 服务的)。

### 3.5.4 样式书写顺序

从本章开始之后的几章会介绍各类样式属性。这些样式属性可以简单地划分为 4 大类,分别是“显示样式”、“自身样式”、“文本样式”和“CSS3 新样式”。

(1) 显示样式: 控制元素展示方式的属性,主要包括浮动(float)、定位(position)、展示方式(display)、超出状态以及可视化(overflow、visibility)等。在第 4、5、6、8 章当中会进行介绍。

(2) 自身样式: 关于元素自身的样式属性,主要就是本章涉及的 5 种属性(宽度 width、高度 height、外边距 margin、内边距 padding、边框 border)以及第 9 章涉及的最大最小宽高。

(3) 文本样式: 用于处理背景图片、段落文章、文字字体的样式。第 7 章中的各类样式均属于文本样式。

(4) CSS3 新样式: CSS3 新增的属性,第 13~16 章当中讲解的属性。

在进行编码时,建议遵循“显示样式”→“自身样式”→“文本样式”→“兼容与 CSS3 新样式”,一方面便于开发者查看,另一方面,浏览器对一个标签样式的解析,是按照“显示样式”→“自身样式”→“文本样式”的顺序执行的。



在此给出比较完整的代码书写顺序(推荐):

- display;
- position;
- position 相关的 left、top、right、bottom、z-index;
- float;
- clear;
- width;
- height;
- margin;
- padding;
- border;
- background;
- color;
- font;
- text-decoration;
- text-align;
- vertical-align;
- white-space;
- text-XXX(其他的 text 类属性);
- CSS3 类。

关于如上的各类样式代码,会在后面的章节当中逐渐讲解到。对于本书当中的各类案例,代码的书写规范也尽可能地遵循了这个顺序。在此需要掌握的是,明确 CSS 的代码书写顺序,并在实际开发当中养成良好的代码编写习惯。

### 3.5.5 CSS 编码规范的问题区

(1) 不遵循编码规范会影响效果实现吗?

大部分情况下都不会。编码规范类似于一个行业标准,是行业人达成的共识,并非浏览器渲染的标准。

开发工程师要掌握编码规范,并在开发中实际应用的主要目的,在于要使自己的代码优秀,从而使自身水平更贴近于行业标准。使用 CSS 给一个标签书写样式并不难,难的是要高质量地实现样式。CSS 最核心的地方在于,如何将代码写得优秀!

(2) 为何不能随便起名字?

类的命名有各种各样的方法,如使用中文类名、汉语拼音、英文单词,各种类名都能够实现想要的样式。

代码实例:

```
<!doctype html >
<html>
<head>
  <meta charset = "UTF - 8">
```

```

<title>HTML5 布局之路 - 各式各样的命名方式</title>
<link rel = "stylesheet" href = "../css/reset.css">
<style>
    .布局之路 {
        width: 200px;
        height: 100px;
        border: 5px solid black;
    }
    .bujuzhilu {
        width: 200px;
        height: 100px;
        border: 1px solid blue;
    }
    .the-road-of-layout {
        width: 100px;
        height: 80px;
        border: 3px solid red;
    }
</style>
</head>
<body>
    <div class = "box">
        <div class = "布局之路">第一个 div</div>
        <div class = "bujuzhilu">第二个 div</div>
        <div class = "the-road-of-layout">第三个 div</div>
    </div>
</body>
</html>

```

显示效果如图 3.11 所示。

对于新手,在进行网站开发时,很容易陷入命名的泥潭当中,虽然各种命名都能够实现,但是在代码量与可维护性方面都存在一定的问题。

使用中文命名,有可能造成编码解读的问题,使用汉语拼音,相对来说比较冗长,而 box1、box2、box3 等让工程师看到类名之后并没有解读出代码段对应的是哪些模块,过长的英语单词会增加代码量,书写以及修改起来都不是很方便,很容易写错,从而出现不必要的问题。

(3) 为何在本书中出现了 con1、con2 等类似的类名?

本书的部分案例使用了 con1、con2 等命名方式,其主要原因是在书写 demo(代码案例),本身标签没有语义性,而在实际的网站开发当中,每个模块都有各自的语义性,不能使用此类命名。因此,无论是本书还是自己进行代码的编写,请区分开测试 demo 与正规开发。

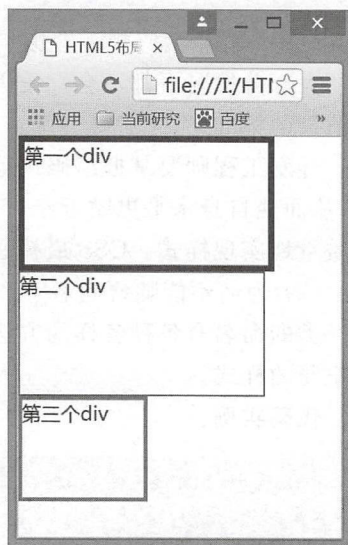


图 3.11 各类 CSS 类名效果图



(4) 关于下画线、连字符、驼峰。

#### ① 三种不同的书写方式。

在命名时,如果需要使用到多个英文单词,书写方式有三种。

假设当前需要为一个列表的标题命名。列表的英文单词是 list,标题的英文单词是 title (可以缩写为 tit)。

书写方式 1:

`.list-tit`。该方法使用连字符进行分隔。

书写方式 2:

`.list_tit`。该方法使用下画线进行分隔。

书写方式 3:

`.listTit`。该方法叫作小驼峰,也就是从第二个单词(含第二个单词)之后的每个单词首字母大写。

#### ② 书写方式的优劣势。

首先,并不建议小驼峰的书写方法,在 JavaScript(行为代码)当中涉及 id 的命名,id 的命名规则为“驼峰命名法”。为了将 id 与类名区分开来,并不会选择驼峰的方式命名类名。

对于连字符和下画线,行业中各有说辞,两者各自有各自的优势。

含义方面:连字符相对更好。从代码含义上来说,连字符是将几个独立单词连接到了一起;下画线,只是在视觉上分隔单词,而在代码含义上来说,类名整体是一个完整的单词。

书写方面:下画线会更方便。从代码的编写方面来说,在编辑器中双击类名,如果是下画线连接的类名,能够选中整个类名;如果是连字符连接的类名,只能够选择到其中的一个类名,如图 3.12 所示。

#### ③ 连字符、下画线,何去何从?

在真正的开发当中,这两者都可以使用。

在当前行业当中,有些公司推荐使用连字符,有些公司推荐使用下画线。根据自己所在公司,

与团队成员保持使用的一致性即可。个人在此更加推荐连字符,相对语义性会更好一些。

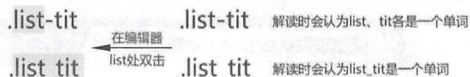


图 3.12 连字符与下画线的区别



## 3.6 CSS 样式重置

### 3.6.1 什么是样式重置

HTML 与 CSS 语言,并不是同期出现的,HTML 要出现的更早一些。那么,在原来没有 CSS 语言的时候,如何控制网页当中的样式呢?

HTML 的发明者在发明标签之时,为这些标签设置了一些样式,这些样式就被称为标签的“默认样式”,“清除标签的默认样式”称为“样式重置”。

### 3.6.2 为何需要样式重置

#### 1. 防止默认样式对 CSS 书写带来的不便

随着时代的发展, CSS 语言的出现, 让开发工程师完全可以通过 CSS 语言控制各个标签的样式, 那么之前标签的默认样式反而成了“累赘”。

例如, 书写一个 h1 标签, 希望它的外边距是 0px, 但是默认情况下 h1 元素存在外边距, 此时还需要额外设置一行代码, 如图 3.13 所示。

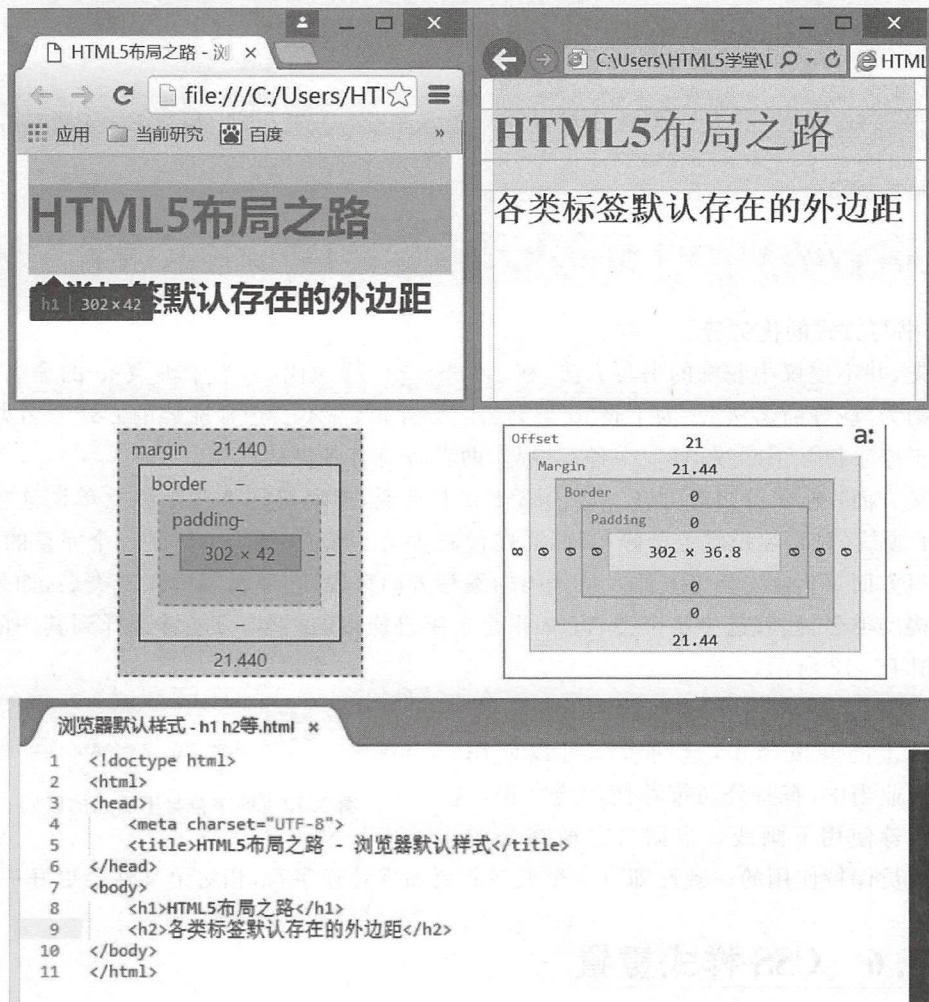


图 3.13 <h1>在各个浏览器中的默认样式

#### 2. 保证网页在各个浏览器表现的一致性

浏览器中样式的不同表现, 也是“清除默认样式”的一个重要因素。在 IE、谷歌、火狐等不同的浏览器中, 同样的标签默认样式并不相同, 如图 3.14 所示。作为开发工程师, 必须保证网页在各个浏览器中表现的一致性, 因此, 需要在开发之前清除掉这种不同的默认样式。



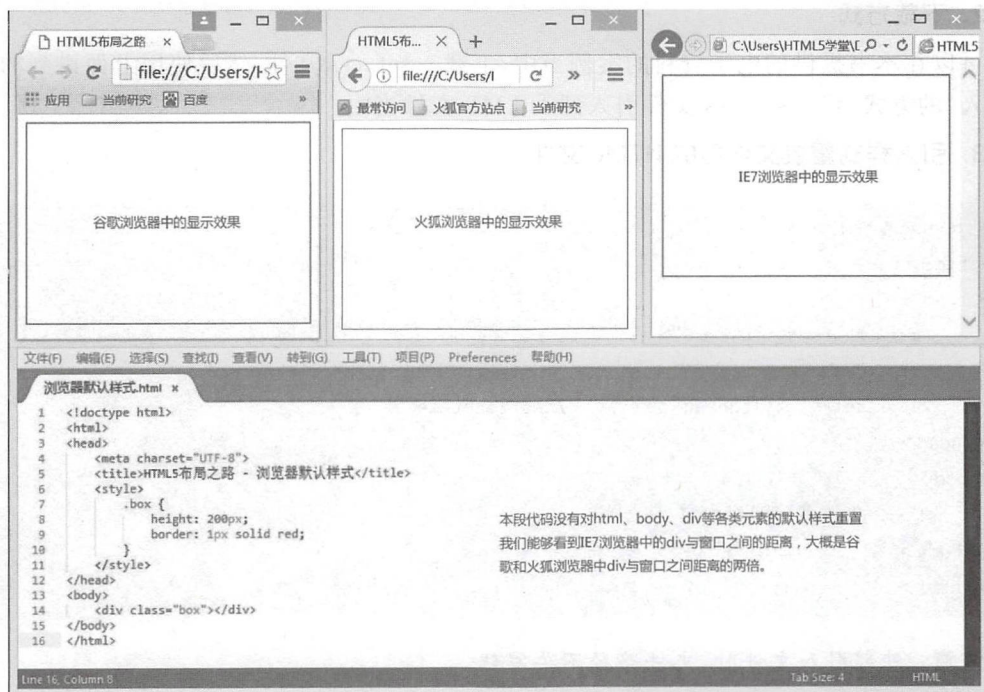


图 3.14 各个浏览器默认样式有所不同

### 3.6.3 样式重置文件

#### 1. 样式重置文件内容

下列代码即清除浏览器各个标签(比较常用的大部分标签)默认样式的代码。

```
@charset 'utf-8';
html{color: #000;background: #FFF;font-family: 'Microsoft YaHei', sans-serif, Arial;}
body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,code,form,fieldset,legend,input,button,
textarea,p,blockquote,th,td,strong{padding:0;margin:0;font-family: 'Microsoft YaHei', sans-
-serif, Arial;}
table{border-collapse:collapse;border-spacing:0;}
img{border:0;}
a{text-decoration:none;color: #333;outline:none;}
a:hover{text-decoration:underline;}
var,em,strong{font-style:normal;}
em,strong,th,var{font-style:inherit;font-weight:inherit;}
li{list-style:none;}
caption,th{text-align:left;}
h1,h2,h3,h4,h5,h6{font-size:100%;font-weight:normal;}
input,button,textarea,select,optgroup,option{font-family:inherit;font-size:inherit;font-
style:inherit;font-weight:inherit;}
input,button,textarea,select{*font-size:100%;}
```

## 2. 下载方式

可以在本书提供的电子资料的案例当中,下载 reset.css 文件,然后使用上面讲到的“外部引入”的方式,将 reset.css 文件引入到 HTML 文件当中。

## 3. 引入样式重置文件后的 HTML 文件

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 引入重置文件后的 HTML 文件</title>
  <link rel = "stylesheet" href = "../css/reset.css">
</head>
<body>
  <!-- 此处书写网页内容 -->
</body>
</html>
```

注意: 外部引入文件时,文件路径不要写错。

## 4. 样式重置文件内容相关解释

在重置文件当中的 CSS 代码,主要有两种功能。一类代码是用来清除掉标签的默认样式,另一类代码则是根据当前的网站制作需求,修改一些默认样式,以方便开发工程师进行网站代码的开发与书写。

关于具体文件当中的内容是什么含义,在学过所有的标签与样式之后,完全可以自己解读。当前请先理解“样式重置的重要性”以及“如何进行样式重置”这两点。



## 3.7 盒模型

### 3.7.1 生活中的“盒模型”——鱼缸

假设一个“卖鱼缸”的商家,当前新进了两个鱼缸,需要存放在库房当中。

库房存放基本需求(俯视状态,单位 cm):

- (1) 鱼缸大小:  $100 \times 100$ 。
- (2) 包装纸箱容量:  $110 \times 110$ 。
- (3) 包装纸箱厚度: 5(4 个方向)。
- (4) 放入纸箱与鱼缸之间泡沫: 5(4 个方向)。
- (5) 纸箱与墙壁、其他纸箱间隔: 5(4 个方向)。

示意图如图 3.15 所示。

### 3.7.2 盒模型基本知识

生活中与代码的盒模型对比如图 3.16 所示。





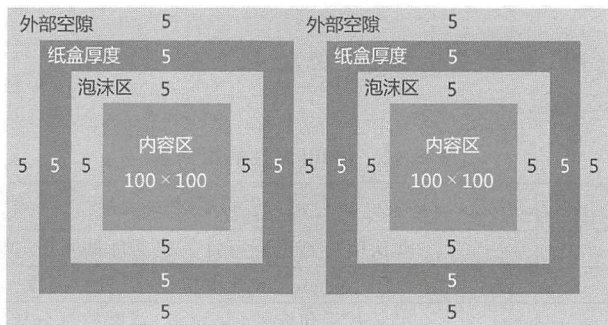


图 3.15 生活中的盒模型

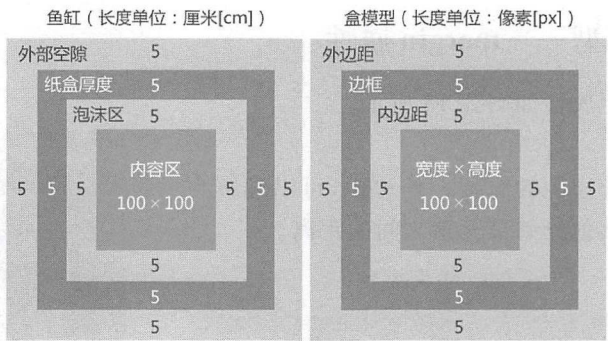


图 3.16 生活中与代码的盒模型对比图

在 HTML 当中,每个元素在浏览器中的解析,都可以被看作一个“盒子”,拥有盒子一样的外形和平面空间。完整的盒模型是由 width、height(宽度和高度构成—>内容)、border(边框)、padding(内边距)、margin(外边距)这几部分属性组成,如表 3.11 所示。

表 3.11 生活与代码中的盒模型对比

鱼 缸	盒 模 型	设置代码
单位: cm	单位: px	width: 100px; height: 100px; margin: 5px; padding: 10px; border: 2px solid #39f;
鱼缸区 100×100	width、height	
泡沫区 10cm	padding	
边界区 2cm	border	
外部空隙区 5cm	margin	

3.7.3 盒模型——width 与 height 属性

盒模型的宽度与高度属性如图 3.17 所示。

属性功能:

width 用于设置元素的宽度; height 用于设置元素的高度。

基本语法:

```
width: 100px;  
height: 100px;
```

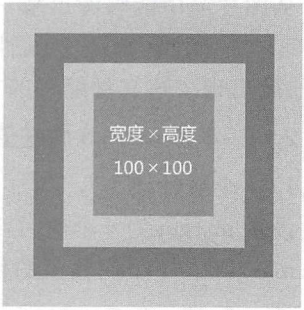


图 3.17 盒模型宽度与高度属性



代码解析：  
设置元素宽度为 100 像素，高度为 100 像素。  
属性值如表 3.12 所示。

表 3.12 width 属性的属性值

值	描 述
auto	默认值。浏览器可计算出实际的宽度/高度
length	使用 px、em、cm 等单位定义宽度/高度
%	定义基于包含块(父元素)宽度的百分比宽度/高度
inherit	规定应该从父元素继承 width/height 属性的值

3.7.4 盒模型——margin 属性

盒模型的外边距属性如图 3.18 所示。

属性功能：  
设置一个元素外边距的宽度。外边距，可以理解为当前元素与父级或其他兄弟级元素之间的距离。

基本语法：

```
margin: 5px 5px 5px 5px;
```

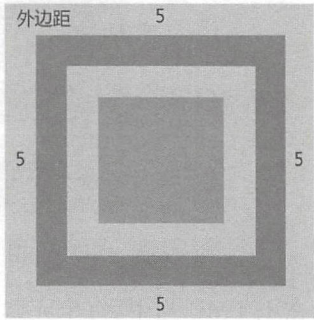


图 3.18 盒模型的外边距属性

代码解析：  
设置元素外边距为 5 像素(4 个方向)。  
属性值(数量：1~4 个)如表 3.13 所示。

表 3.13 margin 属性的属性值

值	描 述
auto	浏览器计算外边距
length	规定以具体单位计的外边距值，如 px、em、cm 等
%	规定基于父元素的宽度的百分比的外边距
inherit	规定应该从父元素继承外边距

1. 为何会有 1~4 个 margin 值

从图片当中能够看出来，一个标签在平面内存在 4 个方向，也就有 4 个外边距值与之相对应。而一个、两个、三个属性值的书写方法，只在于我们希望代码能够书写的更“少”一些。于是，就出现了这三种缩写的方法。

2. 1~4 个 margin 属性值的含义

1) 4 个属性值

设置 margin 为 4 个值时，值与方向的对应顺序为“上-右-下-左”(从顶部，顺时针转下来)。margin 有 4 个属性值的代码：

```
margin: 10px 5px 20px 0px;
```





代码解析：

设置元素的上外边距为 10px，右侧外边距为 5px，下外边距为 20px，左侧外边距为 0px。

2) 三个属性值

设置 margin 为三个值时，值与方向的对应顺序为“上-左右-下”。

margin 有三个属性值的代码：

```
margin: 10px 20px 0px;
```

代码解析：

设置元素的上外边距为 10px，左侧和右侧外边距均为 20px，下外边距为 0px。

3) 两个属性值

设置 margin 为两个值时，值与方向的对应顺序为“上下-左右”。

margin 有两个属性值的代码：

```
margin: 0 20px;
```

代码解析：

设置元素的上、下外边距均为 0px，左侧和右侧外边距均为 20px。

4) 一个属性值

设置 margin 为一个值时，该值表示 4 个方向的外边距均设置为这个属性值。

margin 有一个属性值的代码：

```
margin: 20px;
```

代码解析：

设置元素的上、下、左、右 4 个方向外边距均为 20px。

3. margin 的分写

margin 的分写属性如表 3.14 所示。

表 3.14 margin 分写属性

属 性 名	属 性 含 义
margin-left   right   top   bottom	左侧/右侧/顶部/底部的外边距

该方法书写起来代码较多，使用较少，通常都使用合写方法。

4. margin 的特殊应用

将元素的水平方向 margin 值设置为 auto，能够让块元素在父级当中水平居中。

代码实例：

```
<!doctype html>
<html>
<head>
```



```
<meta charset = "UTF - 8">
<title> HTML5 布局之路 - 元素在父元素中水平居中</title>
<link rel = "stylesheet" href = "../css/reset.css">
<style>
    .box {
        width: 100px;
        height: 100px;
        margin: 0 auto;
        background: # 39f;
    }
</style>
</head>
<body>
    <div class = "box"></div>
</body>
</html>
```

代码解析：

为一个 class(类名)为 box 的元素(div),设置宽度 100 像素、高度 100 像素,上下外边距为 0 像素,左右外边距为 auto(自动),背景颜色为蓝色(#39f)。

3.7.5 盒模型——padding 属性

盒模型的内边距属性如图 3.19 所示。

属性功能：

设置一个元素内边距的宽度。外边距,可以理解为当前元素与元素边框之间的距离。

基本语法：

```
padding: 5px 5px 5px 5px;
```

代码解析：

设置元素内边距为 5 像素(4 个方向)。

属性值(数量: 1~4 个)如表 3.15 所示。

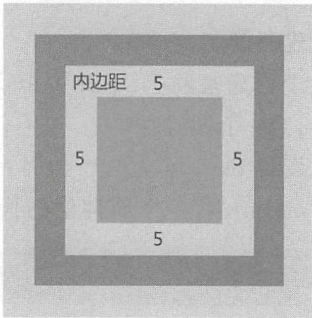


图 3.19 盒模型的内边距属性

表 3.15 padding 属性的属性值

值	描 述
auto	浏览器计算内边距
length	规定以具体单位计的内边距值,比如像素、厘米等。默认值是 0px
%	规定基于父元素的宽度的百分比的内边距
inherit	规定应该从父元素继承内边距

注意：

(1) padding 值与 margin 值类似,都有 1~4 个值,计算的方式方法也相同。





- (2) padding 值也有 padding-left 等 4 个方向的分写方法,分写方式使用很少。
- (3) padding 值并没有负值,设置负值时相当于 0。
- (4) padding 值的单位设置为百分比时,也是按照父级宽度进行计算的。

3.7.6 盒模型——border 属性

盒模型的边框属性如图 3.20 所示。

1. 复合属性

border 是一个复合属性,一个边框包括边框的宽度、边框的颜色以及边框的类型。

边框宽度: 以 px 等为单位。

边框颜色: 十六进制颜色值或单词。

边框类型: solid(实线),dotted(点线),dashed(虚线)。

关于边框的具体属性值后面会进行详细讲解,当前请先了解以上这些知识,然后来看边框的“属性”。

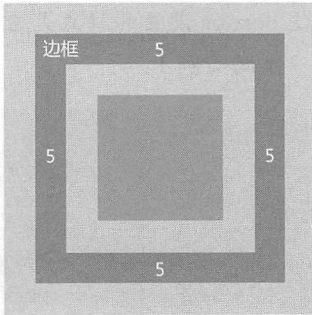


图 3.20 盒模型的边框属性

2. 分写方法

边框有 4 个方向,每个方向的边框都拥有三种属性(宽度、颜色和边框类型),因此就有了如表 3.16 所示的 12 种属性。

表 3.16 边框分写属性的属性值

属 性 名	属 性 含 义
border-left   right   top   bottom-width	左侧/右侧/顶部/底部边框的宽度
border-left   right   top   bottom-style	左侧/右侧/顶部/底部边框的样式
border-left   right   top   bottom-color	左侧/右侧/顶部/底部边框的颜色

border 分写的代码:

```
border-left-width: 10px;
```

3. 缩写方法

看到这里,有没有觉得上面的这种书写方法太冗余了?

于是,关于边框,就出现了如表 3.17 所示的这样几种缩写方法。

表 3.17 边框各属性合写的属性值

属 性 名	属 性 含 义
border-width	设置边框宽度
border-style	设置边框样式
border-color	设置边框颜色
border-left   right   top   bottom	设置左侧/右侧/顶部/底部边框,需要包含边框宽度、边框样式、边框颜色
border	设置 4 个方向的边框宽度、边框样式、边框颜色



**注意：**

(1) border-width、border-style、border-color 均可以取 1~4 个值,属性值与方向的对应关系与“padding、margin”相同。

(2) border-left | top | right | bottom: 需要包含边框宽度、样式、颜色三种属性,不同属性值之间使用空格分隔。

(3) border 与 border-left 等属性类似,均需要三种属性来组合而成,但有所不同的是,通过 border 设置的样式是应用于 4 个方向的。

**代码实例 1:**

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - border 的属性合写 实例 1</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      width: 100px;
      height: 100px;
      padding: 60px;
      border - width: 10px 5px;
      border - style: dotted solid dashed;
      border - color: # 333 # f00 # 0f0 # ff3;
      background: # 39f;
    }
  </style>
</head>
<body>
  <div class = "box">内容</div>
</body>
</html>
```

**代码解析：**

为类名为 box 的 div 设置如下样式：

上下边框宽度 10 像素,左右边框宽度 5 像素；

上边框样式为点线,下边框样式为虚线,左右边框样式为实线；

上边框颜色为深灰色(# 333),右侧边框颜色为红色(# f00),下边框颜色为绿色(# 0f0),左侧边框颜色为黄色(# ff3)。

**代码实例 2:**

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - border 的属性合写 实例 2</title>
```





```
<style>
    .box {
        width: 100px;
        height: 100px;
        padding: 60px;
        border-left: 10px solid #f00;
        background: #39f;
    }
</style>
</head>
<body>
    <div class = "box">内容</div>
</body>
</html>
```

代码解析：

为类名为 box 的 div 设置左侧边框；  
边框宽度 10 像素，颜色红色，实线型边框。

4. border 属性选用原则

border(边框)所涉及的属性多达 20 种，在实际的开发当中，应该如何使用呢？  
一句话来概括：越简单越好，尽可能使用缩写。  
举个例子，当前要为一个元素的左侧、右侧、底部这三个方向均设置“10 像素 实线 红色”的边框，如表 3.18 所示。

表 3.18 border 属性选用方法对比

方法 1 代码与解析	方法 2 代码与解析
方法 1： border-right: 10px solid #f00; border-left: 10px solid #f00; border-bottom: 10px solid #f00;	方法 2：（果断选择这种方法） border: 10px solid #f00; border-top: 0;
代码解析： 使用 border-left 这三种合写方式设置样式，对于没有设置的 border-top，则默认没有边框	代码解析： 首先使用 border 将 4 个方向上均设置好边框，之后再顶部边框清零

5. border 的属性值

在了解 border 的一系列属性之后，下面来具体说说属性值。  
例如：border: 10px solid #f00;

- (1) 10px: 边框宽度，此处可以使用 px，也可以使用 em 的相对度量单位，不允许为百分比，不允许为负值。border 属性的属性值如表 3.19 所示。
- (2) solid: 边框线型的一种，属于“实线”边框。边框类型除了“实线”之外，还包括点线(dotted)、虚线(dashed)。另外，在此罗列各类“线型”，如表 3.20 所示。





表 3.19 border 属性的属性值

值	描 述
thin	定义细的边框
medium	默认。定义中等的边框
thick	定义粗的边框
length	允许自定义边框的宽度
inherit	规定应该从父元素继承边框宽度

表 3.20 边框样式的各类属性值

属性值	具 体 含 义	属性值	具 体 含 义
none	【常用】无边框	double	双线边框
hidden	隐藏边框	groove	3D 凹槽边框
dotted	【常用】点状边框	ridge	3D 垄状边框
dashed	【常用】虚线边框	inset	3D inset 边框
solid	【常用】实线边框	outset	3D outset 边框
inherit	从父级继承边框样式		

(3) #f00: 边框颜色, 此处表示红色。关于颜色, 可以采用多种方式方法来表示, 如表 3.21 所示。

表 3.21 边框颜色的各类表示方法

代 码	颜色 含义	基本格式解释
#ff0000	红色	# 后面为三组十六进制的数字, 第一组表示红色, 第二组表示绿色, 第三组表示蓝色。#ff0000 表示的是红色为 $ff(16 \times 15 + 15 = 255)$ , 绿色和蓝色为 0
#f00	红色	当第 1 和 2 位、第 3 和 4 位、第 5 和 6 位颜色值相同时, 可以进行缩写, 换言之, #f00 等价于 #ff0000 对于 #ffcc89 此类, 有任意两位不同的颜色值, 均不能缩写
red	红色	可以使用英语来表示颜色, 国外很多开发工程师都会使用这种方法, 会比较方便; 但国内使用较少
rgb(255, 0, 0)	红色	rgb 表示的是红绿蓝, 括号中的三个值分别对应于红、绿、蓝, 这三个值是十六进制计算出来的结果, 取值范围为 0~255

3.7.7 盒模型的问题区

温馨提醒: 对于此部分中的问题, 大多数为实战类的问题, 可以在编辑器中动手调试代码, 得出基本结论, 再看答案, 学习方法和基本知识同等重要。

(1) margin: 0 auto 中的 auto 是什么含义?

首先, margin: 0 auto; 这句代码当中, margin 有两个属性值, 第一个属性值表示顶部和底部的外边距, 第二个属性值表示左右的外边距。在 margin: 0 auto; 当中表示横向的外边距自动。此时, 左右外边距的具体的值为: (父级元素内容区宽度 - 含边框内边距的当前元素宽度)/2。





代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - div 元素 margin 设置 auto 值</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      width: 580px;
      height: 250px;
      padding: 0 20px;
      border: 5px solid black;
    }
    .con {
      width: 200px;
      height: 200px;
      margin: 0 auto;
      border: 2px solid black;
    }
  </style>
</head>
<body>
  <div class = "box">
    <div class = "con"></div>
  </div>
</body>
</html>
```

margin 水平方向设置为 auto 时的解析情况如图 3.21 所示。

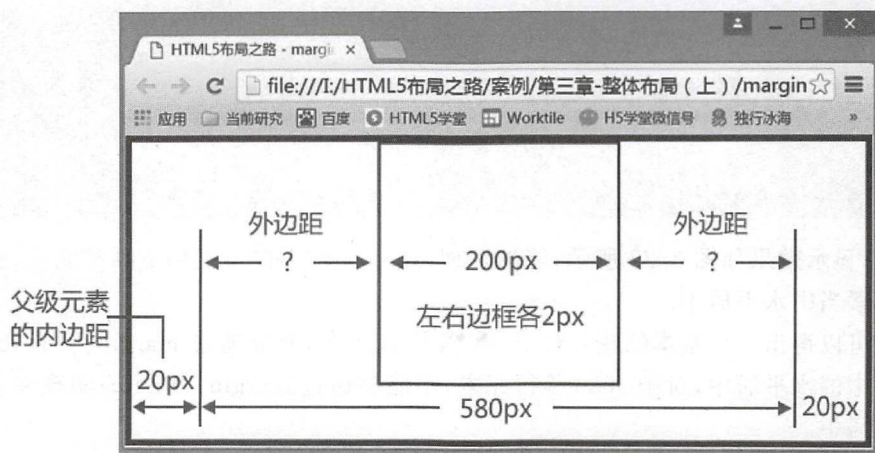


图 3.21 margin 水平方向设置为 auto 时的解析情况

此时子元素(类名为 con 的 div)外边距值是多少呢？

横向 margin 值 =  $580\text{px} - 200\text{px} - 2\text{px} \times 2 = 376\text{px}$ 。



左右 margin 会平分横向剩余的 margin 值,即各为 188px。

(2) 必须设置 margin: 0 auto;才能够实现水平居中吗? margin: 20px auto;之类的命令可以吗?

当然可以! 元素水平居中,是由横向的 margin 值控制的,与纵向 margin 值无关。

(3) 元素水平方向居中,对元素有没有什么限制?

可以尝试将类名为 con 的 div 元素修改为 span 元素,即<span class="con"></span>,之后调试一下效果。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - span 元素的 margin 值</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    .box {
      width: 580px;
      height: 250px;
      padding: 0 20px;
      border: 5px solid black;
    }
    .con {
      width: 200px;
      height: 200px;
      margin: 0 auto;
      border: 2px solid black;
    }
  </style>
</head>
<body>
  <div class="box">
    <span class="con"></span>
  </div>
</body>
</html>
```

此时,显示效果如图 3.22 所示,能够发现,span 元素不仅不支持宽高等属性,也不能够在父级元素当中水平居中。

在此可以得出一个基本结论:对于 div 等块状元素,能够通过 margin: 0 auto;实现在父级元素中的水平居中,对于 span 等行元素,不能够通过 margin: 0 auto;实现在父级元素中的水平居中。

目前只需要使用 div 元素实现布局,对于块元素和行元素,在后面才会逐渐使用到,详见第 5 章。

(4) 如果不为 div 元素设置宽度,但设置了 margin:0 auto;会怎么样?

当 div 没有设定固定宽度时,默认这个 div 会占据父级内容区的 100%。可以认为



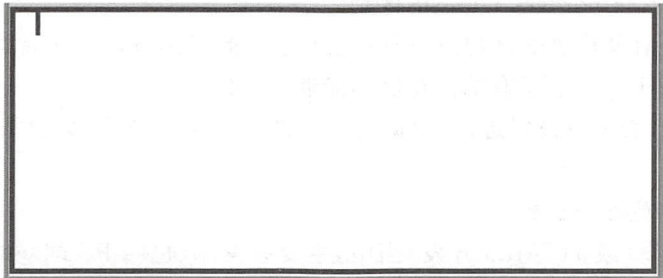


图 3.22 span 元素不能够通过 margin: 0 auto; 水平居中

margin:0 auto;的代码是生效的,只是没有剩余的区域留给 margin 值。

(5) margin 值能否设置为负值?

margin 值可以为负值,而且 margin 负值在模块布局当中也会有很大的用途,在第 6 章当中会详细介绍 margin 负值的应用。在这里,请明确“margin 值可以设定为负值”,如图 3.23 所示。

(6) 设置边框宽度、颜色、样式中的任意两种,会有什么效果?

设置不同边框属性时的边框效果如表 3.22 所示。

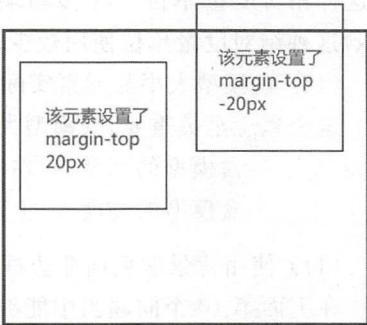


图 3.23 margin 设置负值时的效果

表 3.22 设置不同边框属性时的边框效果

设置编号	边框宽度	边框颜色	边框样式	最终效果
001	设置	不设置	不设置	无效果
002	不设置	设置	不设置	无效果
003	不设置	不设置	设置	有边框效果;默认为黑色,3 像素边框
004	设置	设置	不设置	无效果
005	设置	不设置	设置	有边框效果;默认边框颜色为黑色
006	不设置	设置	设置	有边框效果;默认为黑色,3 像素边框

**基本结论:** 默认的边框颜色为黑色,默认边框粗细为 3 像素(谷歌、火狐、IE8+ 等浏览器下为 3 像素,IE7-浏览器中默认粗细为 4 像素),必须设置“边框样式”属性,才能够触发边框效果。

(7) border、padding、margin 的缩写是否一致?

很多人在最初学习 HTML 时,会混淆盒模型三种属性的缩写。border、padding、margin 具有如下特点。

- ① border、padding、margin 命令均为相应属性的缩写;
- ② 内外边距的属性,仅存在像素大小这种属性值;
- ③ border 属性,包含宽度、样式、颜色三种类型的属性值;
- ④ border 是一个复合属性,border-width、border-color、border-style 的缩写方式和 margin、padding 相同。



(8) border: 0;与 border: none;的区别。

border: 0;表示设置边框,但是边框的宽度为 0,此时浏览器会正常渲染元素的边框效果,会占用内存空间;并且,所有的浏览器均能够正常使用。

border: none;表示不设置边框,浏览器不会进行任何渲染,不会占据内存空间;IE6、7 不兼容。

(9) 盒模型单位如何选择?

当前,前端开发(或 HTML5 开发)当中,主要有桌端(也叫 PC 端)和移动端两个平台,桌端开发的就是台式计算机、笔记本使用的网页,而移动端就是手机使用的网站与页面。

通常,在桌端,像素(px)使用居多,如果涉及响应式、自适应等布局时会考虑使用百分比这种相对度量单位。在移动端,百分比、em、rem 这种相对度量单位使用居多,而像素(px)这种绝对度量单位使用较少。

(10) 盒模型大小与元素实际宽高值(width、height)并不相同。

这个概念至关重要,盒模型大小与元素的实际宽高,并不相同!

盒模型的宽度 = 左右外边距 + 左右边框 + 左右内边距 + width

盒模型的高度 = 上下外边距 + 上下边框 + 上下内边距 + height

(11) 使用背景颜色而非边框标识元素的原因。

在上面第 10 个问题当中能够看出,边框也是构成盒模型的宽度和高度的一部分,如果最初使用边框进行了一个 div 区域的标识,那么在实现模块具体操作之后,这个边框是需要被删除的,此时如果仅删除了边框而没有调整 width 和 height 的大小,就会导致盒模型大小发生变化。一旦开发时忘记调整或调整错误(哪怕是 1 像素的误差),都有可能引起布局错乱。

如果采用背景颜色,并不会占据盒模型的空间大小,再实现页面之后将背景颜色删掉,不需要针对当前元素进行任何其他操作,相对方便简单。

(12) 纵向外边距叠加。

当两个块状元素均设置纵向外边距时,第一个元素的下外边距和第二个元素的上外边距有可能会重合,此时,在默认情况下会出现外边距叠加的现象。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 纵向外边距叠加</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 200px;
      height: 100px;
      margin: 50px;
      background: #39f;
    }
  </style>
</head>
<body>
```





```
<div class = "wrap"></div>
<div class = "wrap"></div>
</body>
</html>
```

显示效果如图 3.24 所示。

#### 代码解析：

第一个 div 有 50 像素的下外边距，第二个 div 有 50 像素的上外边距，此时，两者的外边距重合，只保留 50 像素（如果两个 div 的外边距值不同，则按照较大的那个值来进行计算）。

原因以及解决办法：

之所以产生这个问题，原因在于最早的段落设置。

最初 HTML 中的 p 标签都存在默认的上下外边距，为的是和别的元素之间产生一定的距离。但是用在段落当中时，多个 p 上下排列，每两个 p 元素之间的距离就变得有些大了（上面 p 元素的下外边距+下面 p 元素的上外边距），出于这样的考虑，浏览器解析设置了“纵向外边距叠加”的规则。

解决这个问题并不难，可以为元素设置浮动，纵向外边距就不再叠加（浮动的知识在第 4 章进行讲解）；另外一种方法则是规避掉这个问题，即为元素统一设置上或下，某一个方向的外边距，这样就不会出现重叠的现象和问题了。

（13）父子之间用 padding，兄弟之间用 margin？

行业中有这样一句话：父子之间用 padding，兄弟之间用 margin。这句话具体是什么意思呢？

先来看一个实例，模块布局的需求如图 3.25 所示。

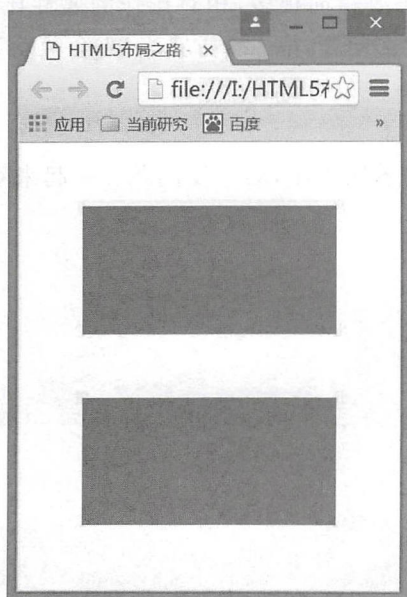


图 3.24 纵向外边距叠加

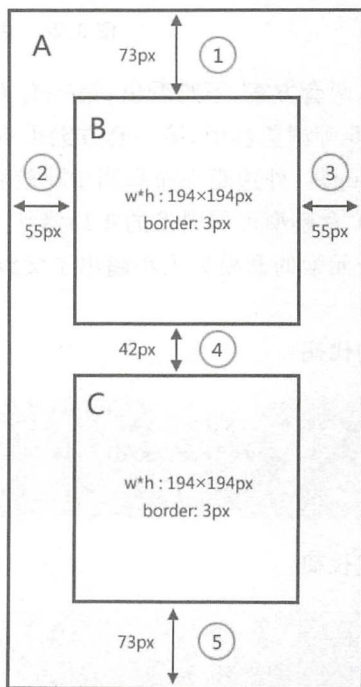


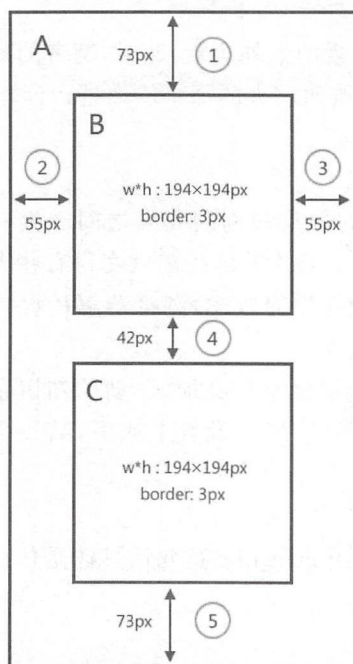
图 3.25 内外边距的选用需求图



图中外部一个 div(A), 里面包含两个 div(B 和 C)。

A、B、C 之间的间距主要包括 1~5 这 5 个地方。其中, ①、②、③、⑤均为父子元素之间的间距, ④为两个同级元素(兄弟元素)之间的间距。

图 3.26 中, 第一套方案遵循了“父子内边距、兄弟外边距”的理念; 第二套方案中采用的都是外边距。



#### 方案1

①②③⑤, 父子元素之间的空隙, 均使用padding值撑开。④, 兄弟级之间的空隙, 使用margin值撑开

样式代码如下:

A元素  
width: 200px;  
padding: 73px 55px;  
border: 3px solid #555;

B元素  
height: 194px;  
margin-bottom: 42px;  
border: 3px solid #555;

C元素  
height: 194px;  
border: 3px solid #555;

#### 方案2

①~⑤, 均使用margin值撑开

样式代码如下:

A元素  
width: 310px;  
border: 3px solid #555;

B元素  
width: 194px;  
height: 194px;  
margin: 73px 55px 42px;  
border: 3px solid #555;

C元素  
width: 194px;  
height: 194px;  
margin: 0px 55px 73px;  
border: 3px solid #555;

图 3.26 内外边距的选用方案对比图

对比两套方案, 不难看出, 第一套方案的代码要精简很多, 相对操作的属性比较少, 另外, 在 IE6 等浏览器中, 第一套方案也不会出现什么问题(第二套方案当中, 会触发 IE6 的双倍边距 bug)。外边距在布局当中比较容易触发一些布局问题。

(14) 盒模型可能引发的布局错乱。

当子元素的盒模型大小超出了父级元素的内容区大小, 会引发问题。一起来看如下这个例子。

结构代码:

```
<div class="box">
  <div class="con"></div>
</div>
```

样式代码:

```
.box {
  width: 200px;
  height: 200px;
```





```

}
.con {
    width: 180px;
    height: 180px;
    margin: 0 auto;
    padding: 10px;
    border: 1px solid #f00;
}

```

#### 代码解析:

“类名为 con 的 div”是“类名为 box 的 div”的子级。

父级 div 的宽度只有 200 像素。

子级的实际宽度为:  $180 + 10 + 10 + 1 + 1 = 202$  像素。

子级的宽度已经超出父级宽度,在这种情况下,在某些浏览器下,很容易引发布局错乱等问题。

在谷歌等容错性比较强的浏览器当中,虽然并不会布局错乱,但是依旧建议修改,防止引发其他问题。

#### (15) 趣味的边框样式。

请思考如图 3.27 所示效果的制作方法。

一个矩形,包含 4 个三角形,4 个三角形颜色各不相同。此处可以使用边框来实现这个效果,通过这个效果,也能够更好地理解边框的显示样式。

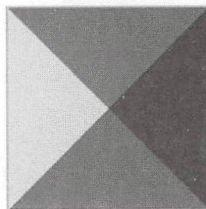


图 3.27 4 个三角形组成的矩形效果

#### 代码实例:

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - border 的特殊效果</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        .box {
            width: 0px;
            height: 0px;
            border - width: 50px;
            border - style: solid;
            border - color: red black green yellow;
        }
    </style>
</head>
<body>
    <div class = "box"></div>
</body>
</html>

```

**代码解析：**

并不为元素设置宽高，所有的部分均由边框来组成，4 个方向各设置了 50 像素不同颜色的实线边框。

边框是从两个角， $45^\circ$ 延伸，最后两条线的延伸线会交叉成一点，如图 3.28 所示。当为 div 元素设置了宽度和高度时（宽高各 100 像素），效果会更加明显。

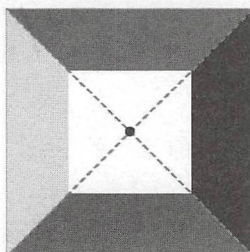


图 3.28 边框的交叉点



## 第4章

# 整体布局(下)——浮动布局



### 4.1 浮动

#### 4.1.1 为何要浮动

在第3章中提到过,div元素属于块元素,块元素默认宽度是占满父级的,如果针对div设置了宽度,这个div的盒模型尺寸虽然成为所设置的大小,但是依旧会“霸占”着整个一行,不让别的元素使用。在网页布局当中,常常需要将块并排进行布局,div标签无法实现这个需求,那么其他元素(标签)能否实现呢?不能!

HTML在最初发明各类元素时,没有几个“容器类标签”既能够设置宽高,又能够和别的元素处于同一行(表格元素除外)。块元素能够设置宽高,但是默认独占一行;行元素默认是内容撑开宽高的,虽然某个行元素能够和其他行元素处于同一行,但是并不能够设置宽高。

此时,如果希望div或其他元素“既能够设置宽高,又能够跟其他元素处于同一行”,就需要使用到浮动。

**备注:**关于块元素、行元素的介绍见第5章。

#### 4.1.2 生活中的“浮动”——水槽

可以把网页看作一个水槽,将需要浮动的元素比作一块积木。如果给元素(积木)设置了float属性,水槽里就有水了,元素(积木)首先应该向上浮起来。如果设置的是float:left;向左浮动,那么元素(积木)从右边向上浮起来到水面,然后向左浮动到水槽的左边。如果给元素设置的是float:right;向右浮动,那么元素先从左边浮起到水面,然后向右浮动到水槽的右边,以此类推。

#### 4.1.3 浮动——float 属性

**属性功能:**

设置一个元素发生浮动,浮动后的元素默认大小为内容大小,并且可以设置宽高,也可以与其他浮动元素或行元素处于同一行。

**基本语法:**

```
float: left;
```

代码解析：  
设置元素浮动，且浮动方向向左。  
属性值如表 4.1 所示。

表 4.1 float 属性的属性值

值	描 述
left	元素向左浮动
right	元素向右浮动
none	默认值。元素不浮动，并会显示其在文本中出现的位置
inherit	规定应该从父元素继承 float 属性的值

4.1.4 浮动特效分析

1. 最简单的浮动效果

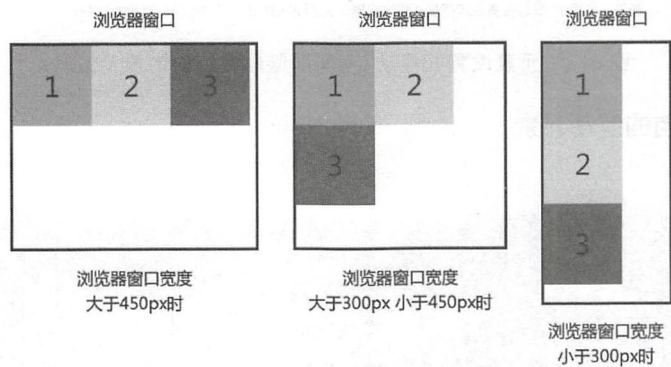
代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 浮动功能案例 1</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box1 {
      float: left;
      width: 150px;
      height: 150px;
      background: #999;
      /* 如下三行代码只是为了便于查看效果 */
      font-size: 40px;
      text-align: center;
      line-height: 150px;
    }
    .box2 {
      float: left;
      width: 150px;
      height: 150px;
      background: #ccc;
      /* 如下三行代码只是为了便于查看效果 */
      font-size: 40px;
      text-align: center;
      line-height: 150px;
    }
    .box3 {
      float: left;
      width: 150px;
      height: 150px;
```



```
background: blue;
/* 如下三行代码只是为了便于查看效果 */
font-size: 40px;
text-align: center;
line-height: 150px;
}
</style>
</head>
<body>
  <div class="wrap">
    <div class="box1">1</div>
    <div class="box2">2</div>
    <div class="box3">3</div>
  </div>
</body>
</html>
```

显示效果如图 4.1 所示。



备注：父级div没有设置固定宽度，其宽度默认为浏览器宽度。子级元素宽高均为150px

图 4.1 元素设置左浮动,不同浏览器宽度时的显示效果

设置左浮动之后,每个块浮动的细化分析如图 4.2 所示。

浏览器窗口大小为380px,三个div宽高均为150px  
三个div均设置了左浮动,图中右下角的数字用于标识子元素的编号,1表示第一个子元素

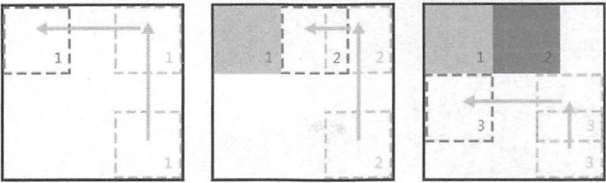


图 4.2 浮动的基本原理

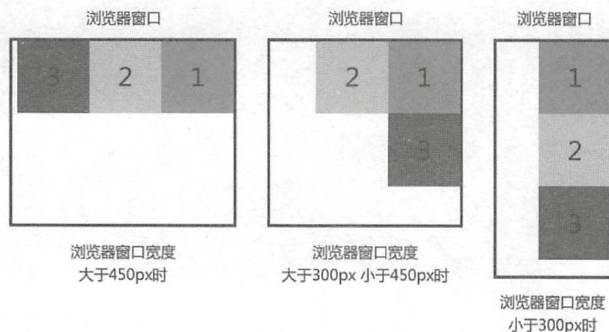
代码解析：

第一个块设置左浮动,基于水槽原理,第一个块从父级容器右下角向上浮起,发现右侧的空余区域大小能够容纳下第一个块,于是浮到最顶端之后,向左漂浮至不能运动为止;

第二个块也遵循了先从下到上,再从右到左的漂浮,与第一个块所不同的是,在向左漂浮的过程中,遇到了第一个块,被第一个块挡在了右侧,于是停止运动。

第三个块依旧遵循先从下到上,再从右到左的漂浮,但是在向上漂时,发现最顶端的空余区域大小并不能够容纳这个块,所以这个块漂到了1和2块下方时,不能够再向上运动,纵向运动停止之后再从右向左漂浮。

如果将上面代码中的“float: left;”修改为“float: right;”显示效果如图 4.3 所示。



备注: 父级div没有设置固定宽度, 其宽度默认为浏览器宽度。子级元素宽均为150px

图 4.3 元素设置右浮动,不同浏览器宽度时的显示效果

## 2. 可能被阻挡的浮动元素

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 浮动功能案例 2</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 300px;
      border: 5px solid black;
    }
    .box1 {
      float: left;
      width: 150px;
      height: 300px;
      background: #999;
      /* 如下三行代码只是为了便于查看效果 */
      font-size: 40px;
      text-align: center;
      line-height: 300px;
    }
    .box2 {
      float: left;
      width: 150px;
```



```

        height: 200px;
        background: #ccc;
        /* 如下三行代码只是为了便于查看效果 */
        font-size: 40px;
        text-align: center;
        line-height: 200px;
    }
    .box3 {
        float: left;
        width: 150px;
        height: 150px;
        background: blue;
        /* 如下三行代码只是为了便于查看效果 */
        font-size: 40px;
        text-align: center;
        line-height: 150px;
    }
    .box4 {
        float: left;
        width: 150px;
        height: 150px;
        background: yellow;
        /* 如下三行代码只是为了便于查看效果 */
        font-size: 40px;
        text-align: center;
        line-height: 150px;
    }
}
</style>
</head>
<body>
    <div class="wrap clearfix">
        <div class="box1">1</div>
        <div class="box2">2</div>
        <div class="box3">3</div>
        <div class="box4">4</div>
    </div>
</body>
</html>

```

显示效果如图 4.4 所示。

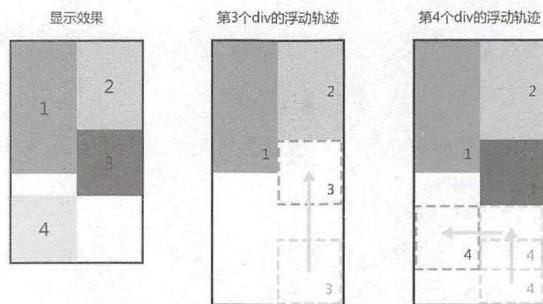


图 4.4 被阻挡的浮动元素案例效果与分析图

#### 代码解析：

4 个子元素宽度均为 150 像素，其中，第一个子元素高度为 300 像素，第二个子元素高度为 200 像素，第三、四个子元素高度为 150 像素。

此处最需要注意的是第三、四个子元素的位置。

第三个子元素从右下向上漂浮，发现在第二个子元素的下面，正好有 150 像素宽度的空间，于是正好进入了那个空白的位置。

第四个子元素从右下向上漂浮，在第三个子元素下方停止运动，之后向左漂浮，到达最终位置。

### 3. 两个方向的浮动

此处两个方向的浮动，并不是指为同一个元素设定“float: left;”和“float: right;”，而是指在同一个父级元素当中，既存在左浮动的元素也存在右浮动的元素。

#### 代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 浮动功能案例 3</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 600px;
      border: 5px solid black;
    }
    .box1 {
      float: left;
      width: 150px;
      height: 150px;
      background: #999;
      /* 如下三行代码只是为了便于查看效果 */
      font-size: 40px;
      text-align: center;
      line-height: 150px;
    }
    .box2 {
      float: left;
      width: 150px;
      height: 300px;
      background: #ccc;
      /* 如下三行代码只是为了便于查看效果 */
      font-size: 40px;
      text-align: center;
      line-height: 300px;
    }
    .box3 {
      float: right;
```



```

width: 150px;
height: 150px;
background: blue;
/* 如下三行代码只是为了便于查看效果 */
font-size: 40px;
text-align: center;
line-height: 150px;
}
.box4 {
float: right;
width: 150px;
height: 200px;
background: yellow;
/* 如下三行代码只是为了便于查看效果 */
font-size: 40px;
text-align: center;
line-height: 200px;
}
</style>
</head>
<body>
<div class="wrap clearfix">
<div class="box1">1</div>
<div class="box2">2</div>
<div class="box3">3</div>
<div class="box4">4</div>
</div>
</body>
</html>

```

显示效果如图 4.5 所示。

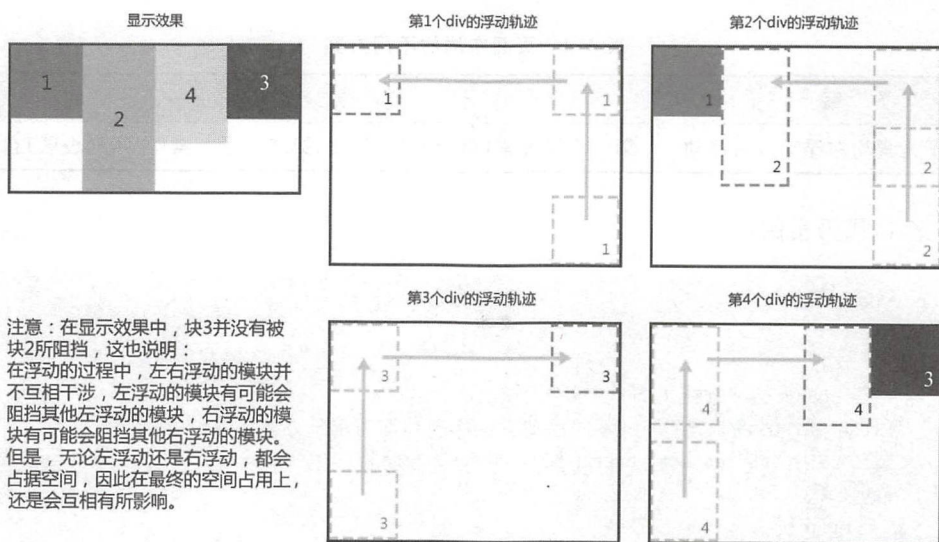


图 4.5 同时存在左右浮动案例效果与分析图

代码解析：

前两个子元素设置左浮动，后两个子元素设置右浮动，在浮动过程当中，左右浮动的元素并不会互相影响、阻挡对方元素。

水槽原理特性小结：

float: left;表示向左浮动，标签(或元素)从右边向上浮起，再从右向左浮动到水槽左边。

float: right;表示向右浮动，标签从左边向上浮起，再从左向右浮动到水槽的右边。

在浮动过程中，如果遇到同方向的其他元素，有可能会被“阻碍”。

在浮动过程中，左浮动的元素和右浮动的元素并不会互相干涉、阻碍对方运动。

在物理空间的占用方面，左右浮动元素会互相影响。

4. 不同的浮动——不同的效果实现思路

功能需求如图 4.6 所示。

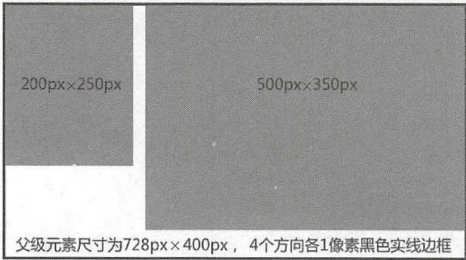


图 4.6 多思路实现浮动效果需求图

父级元素尺寸大小为 728px×400px，4 个方向上各有 1 像素的黑色实线边框，该元素在窗口当中水平居中。在元素当中，包含两个子元素，左侧的子元素是“侧导航”，右侧的子元素是文章列表区域。侧导航部分元素宽高为 200px×250px，文章列表区域的元素宽高为 500px×350px。

实现方案如表 4.2 所示。

表 4.2 页面布局的不同方案

方 案 1	方 案 2
两个子元素均左浮动或右浮动	第一个子元素(侧导航)左浮动,第二个子元素(文章列表区)右浮动

方案 1,代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 不同浮动方法实现功能需求 - 方案 1</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 728px;
```



```

        height: 400px;
        margin: 0 auto;
        border: 1px solid black;
    }
    .nav {
        float: left;
        width: 200px;
        height: 250px;
        margin-right: 28px;
        background: #999;
    }
    .arc-list {
        float: left;
        width: 500px;
        height: 350px;
        background: #ccc;
    }
}
</style>
</head>
<body>
    <div class="wrap">
        <div class="nav">导航区域</div>
        <div class="arc-list">文章列表区域</div>
    </div>
</body>
</html>

```

### 方案 2, 代码实例:

```

<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 不同浮动方法实现功能需求 - 方案 2</title>
    <link rel="stylesheet" href="../css/reset.css">
    <style>
        .wrap {
            width: 728px;
            height: 400px;
            margin: 0 auto;
            border: 1px solid black;
        }
        .nav {
            float: left;
            width: 200px;
            height: 250px;
            background: #999;
        }
        .arc-list {

```



```

        float: right;
        width: 500px;
        height: 350px;
        background: #ccc;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div class="nav">导航区域</div>
        <div class="arc-list">文章列表区域</div>
    </div>
</body>
</html>

```

#### 方案对比：

两种方法均能够实现需求中的效果，相比而言，第二种方法更简便，并不需要额外设置 margin 值，两个元素只需要分别使用左、右浮动，就能够实现效果。

在日常的开发当中，一个网站功能并非只有一种实现方案，不同的构思，不同的设置，都能够达到最终的需求和效果要求，正所谓条条大路通罗马，在众多的道路当中，请不断思考并找出“相对最优秀”的方法。

### 4.1.5 浮动的问题区

为何没有元素能够满足“可设宽高”和“与其他元素同行”的需求？

最早计算机的用途是进行科学计算，主要用于军事方面。网页在最初出现时并没有像现在这么普及，“她”曾经是一个高端产品，主要面向科学家，大量的科学家在网站当中书写科学类的文章，通过网络进行分享。

而后，网络内容逐渐丰富起来，有了搜索引擎，博客微博，各种各样的企业、学校官网，电商平台，以及各类新兴类型的网站与平台。

回顾第 2 章中提到的各类 HTML 标签，不难发现，大致可以分为如下几种。

(1) div、h1、h2、p、ul、ol、li、dl 等：主要用于页面中标题、列表、段落等内容的标识。

(2) u、i、b、code、var、q、blockquote、sub、sup 等：而今的这些标签中，有不少标签已经被弃用了，但是在当时，它们却能够标识出文章中有哪些文字需要加粗、倾斜、添加下画线，哪些文字是上标、下标，哪些是代码，哪些是引文等。

(3) a、img：两种特殊功能的标签，一个是用于放置链接，一个是用来放置图像。

(4) 表格：可以用来让科学家放置研究结果（以表格的方式展示出来）。

(5) 表单类：用于进行用户与机器的交互，以及向后台提交数据。

纵观这些标签，能够感觉到，其实不需要 CSS 代码，各类 HTML 标签足够让科学家们书写文章。那么，论文通常的布局方式是什么样子？是自上而下的排布，还是多个块、段落、标题在浏览器的一行当中显示出来？

很明显，一篇文章，自上而下就好，没有必要让几个段落处于一行来显示。因此，在 HTML 标签最初发明的时候，并没有考虑到这个层面的需求。但是时代发展了，最初对布



局的需求来自文章中的“图文混排”，如何让科学论文更具有可读性？“浮动”就这样华丽地诞生了。之后，网站结构开始变得复杂起来，CSS 样式中的浮动就成了开发必不可少的一部分。



## 4.2 浮动的影响

### 4.2.1 文档流

#### 1. 普通流(文档流)

普通流，也可以称为常规流、文档流。

普通流是文档中可显示对象在排列时所占用的位置。可以将整个网页看作一个文档，这个文档自上而下分成一行一行，并在每行当中按从左至右的顺序，依次排放元素。

#### 2. 脱离文档流

设置浮动的元素，会“脱离文档流”。浮动的元素，并不属于文档中的普通流，元素漂浮于普通流之上，像浮云一样，但能够左右浮动。

由于浮动的这种特性，导致本属于普通流中的元素设置浮动之后，如果“包含框”(父级元素)内部不存在其他普通流元素，也就产生高度为 0 的现象(高度塌陷)，如果“包含框”中存在其他普通流元素，“包含框”的高度依旧有可能因为部分子元素浮动，而受到影响。

出于网站布局的需求，必须要使用浮动才能够实现多个块元素处于同一行，但是浮动会对元素位置造成影响。我们当然不希望“浮动影响布局”的事情发生，所以此处需要掌握“浮动元素对哪些元素会产生影响”，以及“如何闭合浮动元素(也称为清除当前浮动)”，从而使浮动元素的“包含框”(父级元素)表现出正常的高度。

### 4.2.2 浮动元素对父级元素高度的影响

#### 1. 父级元素高度如何计算

在讲解浮动元素对父级元素具体造成什么影响之前，需要先了解父级元素高度的计算方式。

在默认情况下，一个元素的内容大小决定这一个元素的高度(height 值)，当为一个块元素(如 div)设定了高度，那么在主流浏览器当中，这个元素的高度不再由元素内容大小决定，而由设置的这个高度值来决定。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title> HTML5 布局之路 - 父级元素默认高度</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      border: 10px solid #000;
```

```

    }
    .con {
        height: 200px;
        background: #39f;
    }
    .con2 {
        height: 100px;
        background: #f00;
    }
    .box {
        height: 100px;
        border: 10px solid #39f;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div class="con">类名为 wrap 的 div 中的第一个元素</div>
        <div class="con2">类名为 wrap 的 div 中的第二个元素</div>
    </div>
    <div class="box">类名为 box 的 div</div>
</body>
</html>

```

显示效果如图 4.7 所示。

代码解析：

类名为 wrap 的 div，并没有设置任何高度，而是针对该 div 中的两个子元素，设置了高度，分别为 100 像素和 200 像素，在默认情况下，父级 div 由内容撑开高度，即 300 像素。

代码调整——为父级元素设置固定“高度”200px(小于子级元素高度之和)：

```

.wrap {
    height: 200px;
    border: 10px solid #000;
}

```

显示效果如图 4.8 所示。

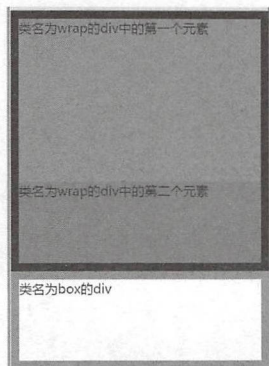


图 4.7 父级元素默认高度由内容撑开

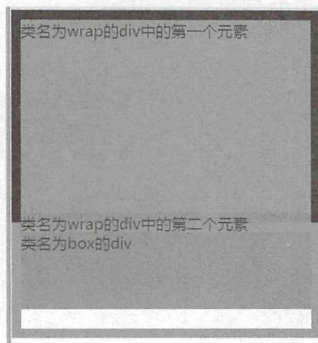


图 4.8 为父级元素设定小于内容区的固定高度



### 代码解析:

浏览器默认内容超出父级的时候依旧显示,因此当我们为父元素设置高度后,父元素按照设置的值进行渲染。内容超出父元素的部分在默认情况下并不会隐藏。

## 2. 子级浮动后,父级元素的高度变化

在页面布局当中,经常会出现这样的现象——父元素本来应该由内容撑开高度,但由于其内部子元素的布局要求,需要对内部的子元素进行浮动。元素一旦浮动,就会脱离文档流,父级元素就相当于“少”了内容,或者说是浮动的这个元素不再对父级的高度产生任何作用或影响。

### 代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 子级浮动对父级元素高度的影响</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    .wrap {
      border: 10px solid #000;
    }
    .con {
      float: left;
      height: 200px;
      background: #39f;
    }
  </style>
</head>
<body>
  <div class="wrap">
    <div class="con">类名为 wrap 的 div 中的第一个元素</div>
  </div>
</body>
</html>
```

115

显示效果如图 4.9 所示。

### 代码解析:

父级的高度原本由内容撑开,当一个子级元素浮动,脱离文档流之后,父级元素的高度受到影响(注意:父级元素默认高度由内容撑开,并不设置固定像素的高度)。

## 3. 必须由内容撑开父级高度的场景

在页面开发当中,的确可以为父级标签添加固定高度,但是有时页面的需求是不固定的高度,即希望内容能够撑开父级元素的高度,比如列表页中的列表部分,内容页的内容区等,如图 4.10 所示。

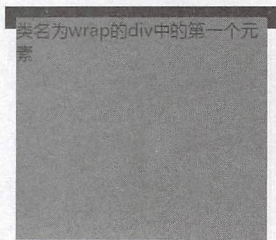


图 4.9 子元素浮动造成父级元素高度塌陷



图 4.10 必须由内容撑开父级高度的场景——列表页

### 4.2.3 浮动元素对兄弟级元素布局的影响

浮动元素,除了对父级元素的高度会产生影响之外,也会对兄弟级元素的布局造成影响。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 浮动元素对其他元素的影响</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      border: 5px solid black;
    }
    .con1 {
      width: 200px;
      height: 100px;
      border: 1px solid blue;
    }
    .con2 {
      float: left;
      width: 100px;
      height: 80px;
      background: # 999;
    }
    .con3 {
      width: 200px;
      height: 100px;
```



```

        border: 1px solid red;
    }
</style>
</head>
<body>
    <div class="box">
        <div class="con1">第一个 div</div>
        <div class="con2">第二个 div, 设置了浮动属性, 具体内容</div>
        <div class="con3">第三个 div</div>
    </div>
</body>
</html>

```

显示效果如图 4.11 所示。

#### 代码解析:

在一个父级元素当中存在着三个 div 元素, 第二个 div 元素设置了浮动并设置了宽度和高度, 第一个和第三个 div 元素并没有设置浮动, 这时, 第三个元素的布局会受到第二个浮动元素的影响。

从具体效果也能够看出: 浮动元素会对兄弟级元素造成影响, 但是仅针对它后面的兄弟级元素, 并不会对前面的兄弟级元素的布局造成影响。

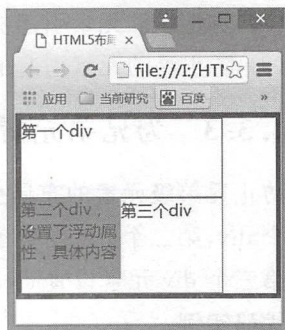


图 4.11 浮动元素对兄弟级元素布局的影响



## 4.3 清除浮动

### 4.3.1 浮动——clear 属性

#### 属性功能:

用于清除浮动, 规定元素的哪一侧不允许存在其他浮动元素。

#### 基本语法:

```
clear: both;
```

#### 代码解析:

设置元素左右两侧均不允许存在浮动元素。

属性值如表 4.3 所示。

当既希望实现子元素的浮动布局, 又希望父级元素由内容撑开高度时; 或既希望实现元素的浮动布局, 又不希望浮动元素对其他兄弟级元素产生影响时, 就需要针对浮动元素或相关元素进行一些操作, 这些操作的主要目的是“清除掉浮动元素对其他元素的影响”。

**注意:** 清除浮动, 并不是把浮动元素删掉, 而是取消掉“浮动元素浮动效果”对其他元素造成的影响。



表 4.3 clear 属性的属性值

值	描 述
left	在左侧不允许浮动元素
right	在右侧不允许浮动元素
both	在左右两侧均不允许浮动元素
none	默认值。允许浮动元素出现在两侧
inherit	规定应该从父元素继承 clear 属性的值

4.3.2 清除浮动的不同类型

根据清除浮动的目的,可以分为以下两大类。

- (1) 防止浮动元素引起的兄弟级元素布局受到影响。
- (2) 防止浮动元素引起的父级高度塌陷。

4.3.3 为兄弟元素设置 clear 样式

防止兄弟级元素的布局受到影响的方法,相对就要简单一些。假设父级元素当中存在着三个 div,第二个 div 发生了浮动,这个时候第三个 div 元素的布局有可能受到影响,只需要为第三个 div 元素设置 clear 属性即可(clear 属性使用的具体属性值根据情况而定)。

代码实例:

```
<!doctype html >
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 为兄弟元素设置清浮动</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      border: 5px solid black;
    }
    .con1 {
      width: 200px;
      height: 100px;
      border: 1px solid blue;
    }
    .con2 {
      float: left;
      width: 100px;
      height: 80px;
      background: #999;
    }
    .con3 {
      clear: both;
      width: 200px;
      height: 100px;
```





```

        margin-top: 20px;
        border: 1px solid red;
    }
</style>
</head>
<body>
    <div class="box">
        <div class="con1">第一个 div</div>
        <div class="con2">第二个 div, 设置了浮动属性, 具体内容</div>
        <div class="con3">第三个 div, 设置了清除浮动属性</div>
    </div>
</body>
</html>

```

显示效果如图 4.12 所示。

#### 代码解析:

第二个子元素发生了浮动, 此时, 第三个子元素的布局会受到影响, 因此在第三个元素的样式当中, 进行清除浮动的处理, 从而去除浮动元素的影响。

此外, 由于第三个子元素进行了清浮动, 因此, 父级的高度塌陷问题也就此解决。

此处需要注意两点:

(1) 并非浮动元素的所有兄弟级元素都需要清除浮动, 只需要针对浮动元素的下一个兄弟级元素设置清浮动, 后面所有元素的布局都会恢复。

(2) 如果希望在第二个 div(浮动元素)与第三个 div(清除浮动的兄弟级元素)之间有一定的间距, 为浮动元素后的一个兄弟级元素设置顶部外边距时会失效(与上方空白区叠加), 此时, 可以为浮动元素设置下边距。

margin 的失效问题:

在上面的案例中, 为第三个 div 设置了清除浮动和顶部的外边距, 但是会发现, 第三个 div 并没有和第二个 div 分开, 第三个 div 的 margin 值与第二个 div 的内容区域产生重合, 如图 4.13 所示。

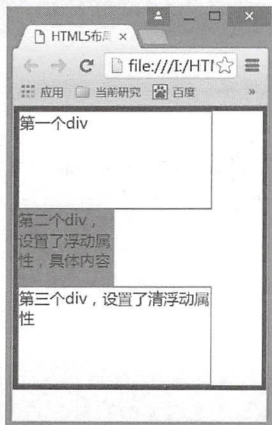


图 4.12 清除浮动元素对兄弟级元素布局的影响

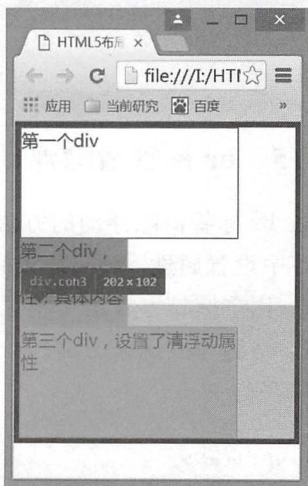


图 4.13 清浮动后兄弟级元素 margin 失效的情况



#### 4.3.4 空标签清除浮动

操作：在浮动元素的后面，增加了一个新标签，这个新标签是浮动元素的兄弟级元素，之后为这个标签设置 clear 属性。

优点：通俗易懂，操作方便。

缺点：会添加大量无语义空标签，结构与表现未分离，不利于维护。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 空标签清除浮动</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      border: 10px solid #000;
    }
    .con {
      float: left;
      height: 200px;
      background: #39f;
    }
    .clear {
      clear: both;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "con">类名为 wrap 的 div 中的第一个元素</div>
    <div class = "clear"></div>
  </div>
</body>
</html>
```

#### 4.3.5 br 标签清除浮动

操作：br 标签清除浮动的方法类似空标签清除浮动，在浮动元素后面添加一个 br 标签，在 br 标签中设置属性 clear，并赋值 all，即：<br clear="all">。

优点：比空标签方式语义稍强，代码量较少。

缺点：结构与表现未分离。

代码实例：

```
<!doctype html>
<html>
```





```

<head>
  <meta charset = "UTF - 8">
  <title> HTML5 布局之路 - br 标签清浮动</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      border: 10px solid #000;
    }
    .con {
      float: left;
      height: 200px;
      background: #39f;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "con">类名为 wrap 的 div 中的第一个元素</div>
    <br clear = "all">
  </div>
</body>
</html>

```

### 4.3.6 父元素浮动

操作：为当前浮动元素的父级元素设置浮动。

优点：语义化没问题,同时代码量少。

缺点：父元素的相邻元素布局受影响——于是需要继续为其父级的父级设置浮动操作,直到 body 为止。

代码实例：

```

<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title> HTML5 布局之路 - 父元素浮动</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      float: left;
      border: 10px solid #000;
    }
    .con {
      float: left;
      height: 200px;
      background: #39f;
    }
  </style>

```



```
</head>
<body>
  <div class = "wrap">
    <div class = "con">类名为 wrap 的 div 中的第一个元素</div>
  </div>
</body>
</html>
```

#### 4.3.7 父元素设置 overflow: hidden 或 auto

操作：为当前浮动元素的父级元素设置 overflow: hidden 或 auto。

优点：语义化没问题，同时代码量少。

缺点：内容多的时候，会被隐藏，无法显示需要溢出的元素，也可能会对之后 JS 的一些动态效果操作造成影响。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - overflow 清浮动</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      overflow: hidden;
      border: 10px solid #000;
    }
    .con {
      float: left;
      height: 200px;
      background: #39f;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "con">类名为 wrap 的 div 中的第一个元素</div>
  </div>
</body>
</html>
```

备注：关于 overflow，在第 6 章当中会详细讲解。

#### 4.3.8 利用 after 伪元素清除浮动

操作：为当前浮动元素的父级元素添加 after 伪元素，为 after 伪元素设置清除浮动的功能代码。





优点：语义化和结构都没有问题。

缺点：如果使用不合理，有可能造成代码量增加；另外，IE6/7 不支持 after 伪元素，需要使用 zoom:1 触发 hasLayout 来清浮动。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 伪元素清浮动</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      border: 10px solid #000;
    }
    .con {
      float: left;
      height: 200px;
      background: #39f;
    }
    .wrap:after {
      content: '\200B';
      clear: both;
      display: block;
      height: 0;
    }
    .wrap {
      * zoom: 1;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "con">类名为 wrap 的 div 中的第一个元素</div>
  </div>
</body>
</html>
```

代码解析：

after 伪元素清浮动的原理其实与空标签清浮动的原理类似。

content: '\200B'; 中 content 表示的是内容，对于伪元素来说，如果没有内容，则不会显示在页面当中，因此需要设置内容，但是如果将内容设置出来，就会展示在页面上。在此只是想借助 after 伪元素来清浮动，自然不希望内容显示在页面当中，于是使用\200B，它表示一个零宽度的空格。因此，这句代码就相当于为 after 伪元素定义了内容，使 after 伪元素能够显示出来；又将内容设置成了零宽度的空格，让这个特殊的内容不影响页面布局 and 显示。

display: block; 代码是将伪元素设置为“块状元素”，也就是让其具备 div 元素的特点。



height: 0; 代码是将伪元素的高度设置为 0, 目的在于防止高度对布局造成的影响。

### 4.3.9 after 伪元素清除浮动的实际用法

#### 1. 统一书写

由于清除浮动会广泛地存在于网页开发中, 因此为了不书写大量的重复代码, 会将 after 伪元素清除浮动的代码单独使用一个类名来进行设置, 然后在多个地方使用。

CSS 代码设置:

```
.clearfix:after {
    content: '\200B';
    clear: both;
    display: block;
    height: 0;
}
.clearfix {
    * zoom: 1;
}
```

#### 2. 类的组合

对于一个 HTML 中的结构元素, 一方面具备自己的样式, 另一方面还具备一些公共样式, 这时如何书写呢?

样式代码:

```
.common {
    border: 1px solid red;
}
.subnav {
    float: left;
    width: 200px;
}
.con {
    float: right;
    width: 800px;
}
```

结构代码:

```
<body>
  <div class="common subnav">第一个 div</div>
  <div class="common con">第二个 div</div>
</body>
```

代码解析:

第一个 div 渲染后的样式为: 左浮动、200 像素宽度, 1 像素的红色实线边框。

第二个 div 渲染后的样式为: 右浮动、800 像素宽度, 1 像素的红色实线边框。





两个 div 都存在 1 像素的红色实线边框,这种类名的书写方式就是“类的组合”。让标签既加载 common 选择器设置的样式,又加载 subnav 或 con 选择器设置的样式。

class 是标签的属性,用于表示为元素设置类名,而等号后面的是具体的类名。这时候可以为一个元素设置多个类,不同的类名使用空格进行分隔即可,一个元素可以有一个或多个类,并无限制。

对于 after 伪元素清浮动的应用,也可以如法炮制。在上面的代码中已经将 after 伪元素清浮动的代码独立了出来,即,clearfix{}。那么在网页开发时,将这段代码放置到 CSS 文件中,之后在布局时,如果需要使用,就利用类名的组合进行书写即可。

注：此处 clearfix 只是一个类名，也可以自定义其他类名。

#### 4.4 关于“清除浮动”的问题区

#### 4.4.1 clear:both 的兼容问题

清除浮动,很多人一直都在用 `clear:both`,但是很少人了解这个属性的兼容问题。在 IE6 和 IE7 浏览器当中,是不兼容 `clear:both` 的。处理兼容的方法是使用 hack 技术在样式代码中增加 `*float:none`;这句代码。(在第 12 章当中会进行 hack 技术的详细讲解。)

#### 4.4.2 为父级设置高度是不是清除浮动的方法

元素浮动之后,会导致父级高度塌陷,此时可以为父级设置高度,这会使父级元素的高度恢复正常,但是这并不代表清除掉了浮动。

因此,为父级设置高度,并不是一种清除浮动的方法,只是一种防止父级布局受到影响的方法。

#### 4.4.3 浮动元素与非浮动元素处于同一行时会出现的问题

代码实例：

```
<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - 浮动与非浮动元素处于同一行</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        .box {
            width: 400px;
            height: 100px;
            border: 2px solid black;
        }
        .box span {
            float: left;
        }
    </style>

```

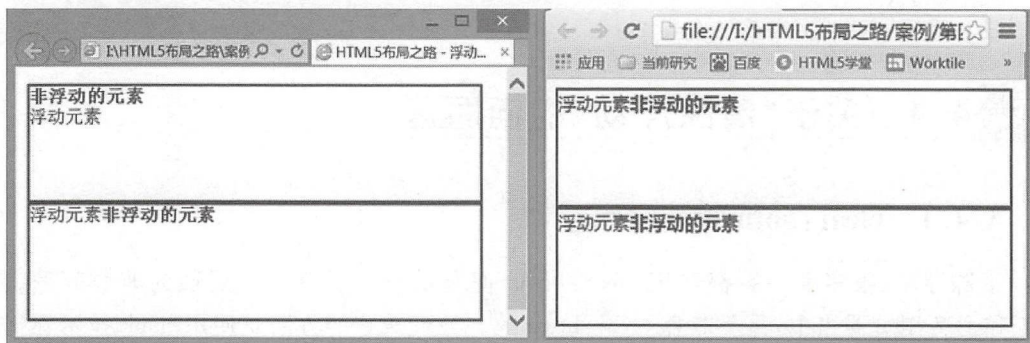


```

</style>
</head>
<body>
  <div class="box"><strong>非浮动的元素</strong><span>浮动元素</span></div>
  <div class="box"><span>浮动元素</span><strong>非浮动的元素</strong></div>
</body>
</html>

```

显示效果如图 4.14 所示。



IE7浏览器 显示效果

谷歌浏览器 显示效果

非浮动元素与浮动元素处于同一行时  
有可能会发生换行

非浮动元素与浮动元素处于同一行时  
元素的顺序会发生变化

图 4.14 浮动元素与非浮动元素处于同一行的显示效果

代码解析：

在 IE8+ 以及其他主流浏览器当中，当浮动元素与非浮动元素处于同一行时，浮动元素在前，非浮动元素在后；对于 IE6、IE7 浏览器，当非浮动元素在前，浮动元素在后时，会产生换行。此处的问题仅针对行元素。

#### 4.4.4 清除浮动方法的选择

浮动之后，有各种方法实现清除浮动，这些方法当中，哪个方法会更为合适呢？

通常情况下，使用最为频繁的是 after 伪元素清浮动，它是空标签清浮动的“升级版”，该方法能够让代码量变得更简单，更便于进行操作。



## 4.5 较为复杂的浮动布局

在稍复杂的网页布局当中，并不局限于一层结构，可以采用多层次结构，针对相应标签设置浮动等属性，从而实现更佳的效果。

### 4.5.1 功能需求

较复杂浮动布局的功能需求如图 4.15 所示。





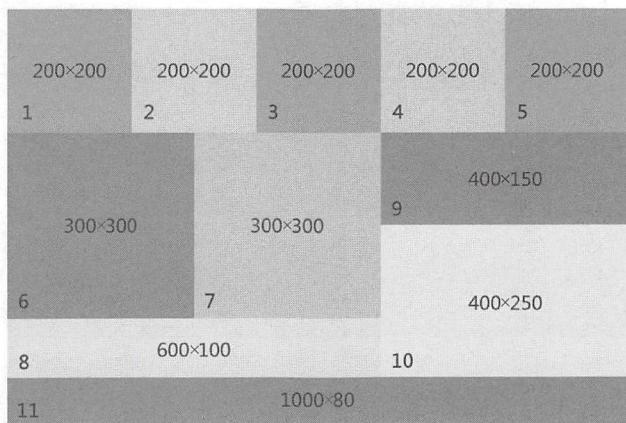


图 4.15 较复杂浮动布局的功能需求

备注：针对不同的块使用了不同的背景颜色标识了出来。另外，每个块左下角的数字是针对块的一个编号；此外，在代码书写时，在类名部分也采用了与每个模块编号相对应的序号，这样的操作主要是为了方便代码的讲解，与实际功能需求并没有关系。

较复杂浮动布局的需求分析如图 4.16 所示。

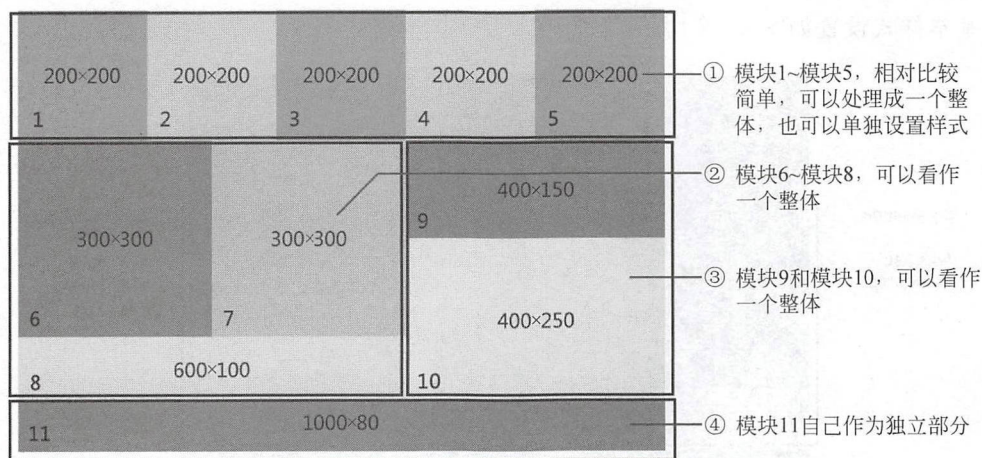


图 4.16 较复杂浮动布局的需求分析

## 4.5.2 需求分析

结构代码：

```
<div class = "wrap">
  <div class = "box1"> 1 </div>
  <div class = "box2"> 2 </div>
  <div class = "box3"> 3 </div>
  <div class = "box4"> 4 </div>
  <div class = "box5"> 5 </div>
  <div class = "left">
```



```
<div class = "box6">6</div>
<div class = "box7">7</div>
<div class = "box8">8</div>
</div>
<div class = "right">
  <div class = "box9">9</div>
  <div class = "box10">10</div>
</div>
<div class = "box11">11</div>
</div>
```

需求解析：

将 6~8 设置为一个左侧模块,9 和 10 设置为一个右侧模块,这种设置方式会让布局变得更加方便。首先将左侧模块设置左浮动,右侧模块设置右浮动。之后在左侧模块当中,6 和 7 分别进行浮动,8 模块默认占满父级,原则上来说,只需要设置高度即可,但是由于 6 和 7 进行了浮动,会对 8 模块的布局造成影响,此处需要针对模块 8 清除浮动。

对于右侧模块,9 和 10 并不需要发生任何浮动,只需要设置高度即可。

对于第 11 模块,由于前面包含 6~8、10 和 11 的两个大模块发生了浮动,也会对该模块造成影响,因此,需要针对第 11 个模块进行浮动的清除。

基本样式设置如图 4.17 所示。

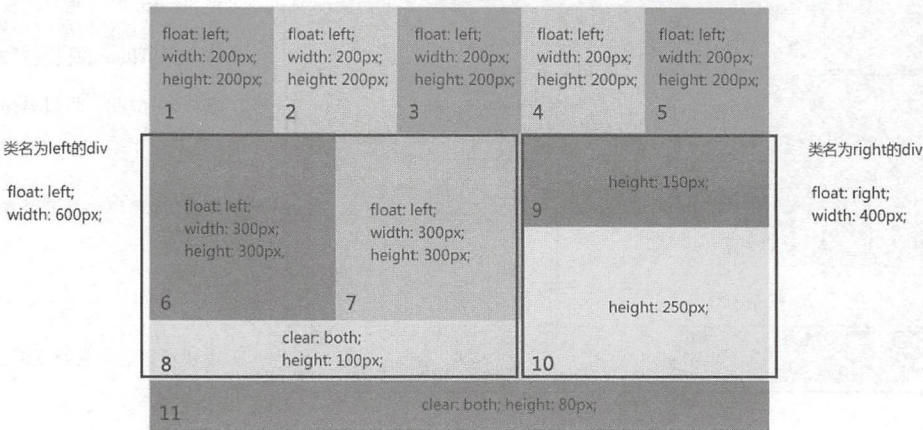


图 4.17 较复杂浮动布局的基本样式设置

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 较为复杂的浮动布局案例</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
```





```
        width: 1000px;
        margin: 0 auto;
    }
    .common {
        float: left;
        width: 200px;
        height: 200px;
    }
    .left {
        float: left;
        width: 600px;
    }
    .right {
        float: right;
        width: 400px;
    }
    .left-common {
        float: left;
        width: 300px;
        height: 300px;
    }
    .box8 {
        clear: both;
        height: 100px;
    }
    .box9 {
        height: 150px;
    }
    .box10 {
        height: 250px;
    }
    .box11 {
        clear: both;
        height: 80px;
    }
}
```

/\* 如下代码只是控制背景颜色,后期会删除,出于代码篇幅考虑,如下代码采用单行形式进行书写 \*/

```
.box1 {background: #ccc;}
.box2 {background: #a9a9a9;}
.box3 {background: #ccc;}
.box4 {background: #a9a9a9;}
.box5 {background: #ccc;}
.box6 {background: #999;}
.box7 {background: #d9d9d9;}
.box8 {background: #ccc;}
.box9 {background: #999;}
.box10 {background: #a9a9a9;}
.box11 {background: #d9d9d9;}
```

</style>

</head>



```
<body>
  <div class = "wrap">
    <div class = "common box1">1</div>
    <div class = "common box2">2</div>
    <div class = "common box3">3</div>
    <div class = "common box4">4</div>
    <div class = "common box5">5</div>
    <div class = "left">
      <div class = "left - common box6">6</div>
      <div class = "left - common box7">7</div>
      <div class = "box8">8</div>
    </div>
    <div class = "right">
      <div class = "box9">9</div>
      <div class = "box10">10</div>
    </div>
    <div class = "box11">11</div>
  </div>
</body>
</html>
```

备注：在这段代码当中，box1~box7 的类名，只是为了能够显示背景颜色，在真正开发时会被删掉。





## 第5章

# 模块布局(上)——选择标签



### 5.1 为何要选择标签

在实际开发当中,一个网页的布局,存在着多种实现方法,从实现的角度来说,不论选用 div、dl 还是 table 标签进行布局,它们都能够达到我们的要求。然而,实现基本布局是网站开发的“及格线”,对于开发工程师来说,要在各种方法当中选择一种较好的方法。

考量一种实现方法的优劣,主要包括这样几个因素:页面代码精简度、是否符合开发规范、SEO(标签语义性)、扩展性等。

**注意:**在网站开发当中,并非所有的标签都可以被替代。上面提到的可以被替代的标签通常指的是不涉及特殊功能的标签,比如 div、p 等。对于 a 标签、img 标签、form 表单等拥有特定功能的标签,要保证功能的实现,必然是不能随意替换的。



### 5.2 开发时可以选用的标签以及功能

在本节当中,会详细介绍模块布局涉及的各类标签,如 h1、p、ol、dl 等;对于表格、表单、超链接 a 标签、图像 img 标签等特殊功能用的标签,会在后面相应部分进行详细讲解。之所以如此安排,也是为了按照网站开发步骤逐步推进。

在本节的案例代码当中,代码均为< body>元素中的代码,且 HTML 文件不要引入重置样式文件,否则无法看到浏览器默认的显示效果。

#### 5.2.1 h1~h6 标题类标签

**标签含义:**

h1~h6 的标签都属于标题类标签,分别表示不同级别的标题,< h1 >定义最大的标题,< h6 >定义最小的标题。

在浏览器默认状态下,每种标题均为加粗效果,且各自对应不同的文字大小,h1 为 2em,h2 为 1.5em,h3 为 1.17em,h4 为 1em,h5 为 0.83em,h6 为 0.67em。(此处小数点精确到了第二位,em 表示字符大小,浏览器默认字体大小为 16 像素,1em=16px;如果字体大小设置为 20px,那么 1em=20px。)



代码范例：

```
<h1>h1 代表标题 1 </h1>
<h2>h2 代表标题 2 </h2>
<h3>h3 代表标题 3 </h3>
<h4>h4 代表标题 4 </h4>
<h5>h5 代表标题 5 </h5>
<h6>h6 代表标题 6 </h6>
```

### 5.2.2 hr 分隔线

标签含义：

hr 标签表示分隔线，会以一条直线的方式进行展示。hr 标签是一种单标签，即只需要书写成< hr>即可。

分隔线默认占满父级的整行。

代码范例：

```
<h6>h6 代表标题 6 </h6>
<hr>
<p>具体段落内容</p>
```

### 5.2.3 p 与 br 段落与换行

标签含义：

p 标签用于表示一个段落。

br 标签表示换行，为单标签，通常出现在 p 标签当中。

每个 p 标签就是一个段落，而使用 br 标签换行，通常被称为软换行，即虽然使用 br 之后，段落的文本内容从一个新行开始显示，但是这些文本依旧是一个段落。

代码范例：

```
<p>具体段落内容</p>
<p>具体<br>段落内容</p>
```

### 5.2.4 无序列表与有序列表

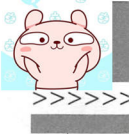
标签含义：

ul 表示无序列表，该种列表中的每个列表项前面，在浏览器中默认显示为一个圆形的黑点●。

ol 表示有序列表，该种列表中的每个列表项前面，在浏览器中默认显示为数字类型的序号 1、2 等。

无论是 ul 还是 ol，列表中每个具体的列表项都使用的是 li 标签。





代码范例：

```
<ul>
  <li>无序列表项 01</li>
  <li>无序列表项 02</li>
  <li>无序列表项 03</li>
</ul>
<ol>
  <li>有序列表项 01</li>
  <li>有序列表项 02</li>
  <li>有序列表项 03</li>
</ol>
```

列表的项目符号类型与位置：

可以通过 list-style-type, 来为列表元素 (ul 或 ol) 设置具体的项目符号类型。较为常见的符号类型包括：

disc(实心圆)、circle(空心圆)、decimal(阿拉伯数字)、lower-alpha(小写英文字母)、upper-alpha(大写英文字母)、upper-roman(大写罗马数字)、lower-roman(小写罗马数字)、none(不使用项目符号)。

可以通过 list-style-position 来设置项目符号的显示位置, 该属性包含 inside 和 outside 两种属性值, 分别表示在列表项文本行内、行外来显示项目符号。

### 5.2.5 自定义列表

标签含义：

自定义列表包括 dl、dt、dd 三种标签。其中, dl 表示自定义列表, dt 表示自定义列表的标题, 而 dd 表示自定义列表的内容。每个 dl 当中, 原则上只能够出现一个 dt, 可以出现多个 dd。

代码范例：

```
<dl>
  <dt>自定义列表的标题</dt>
  <dd>自定义列表的内容</dd>
  <dd>
    <div>自定义列表内容中可以嵌套其他块元素</div>
  </dd>
</dl>
```

### 5.2.6 行内标签

如下的这些标签均属于行内标签, 都可以应用于某一个块元素之内, 均能够与其他行内元素处于同一行, 这些标签的主要目的在于实现行内模块的样式控制。

标签含义：

span, 没有任何特殊含义, 默认状态下没有特殊样式。

em, 表示强调, 默认状态下为倾斜效果。

strong, 表示强调, 默认状态下为加粗效果。

var, 表示变量, 默认状态下为倾斜效果。



b,表示加粗,默认状态下为加粗效果。  
i,表示倾斜,默认状态下为倾斜效果。  
u,表示下画线,默认状态下文本有下画线样式。  
代码范例:

```
<div>
  <span> span 元素</span>
  <em> em 元素</em>
  <strong> strong 元素</strong>
  <var> var 元素</var>
  <b>b 元素</b>
  <i>i 元素</i>
  <u>u 元素</u>
</div>
```

额外注意:  
在这 7 种标签当中,span、em、strong 元素使用的较多,后面 4 种标签在布局标签不够用时才会派上用场。当然,除了这 7 种标签之外,还有各种能够辅助布局的行内元素,由于使用较少,在此就不做讲解了。

5.2.7 代码范例的显示效果图

可将前述的各类标签书写到一起,形成 HTML 文件在浏览器中查看,显示效果如图 5.1 所示。

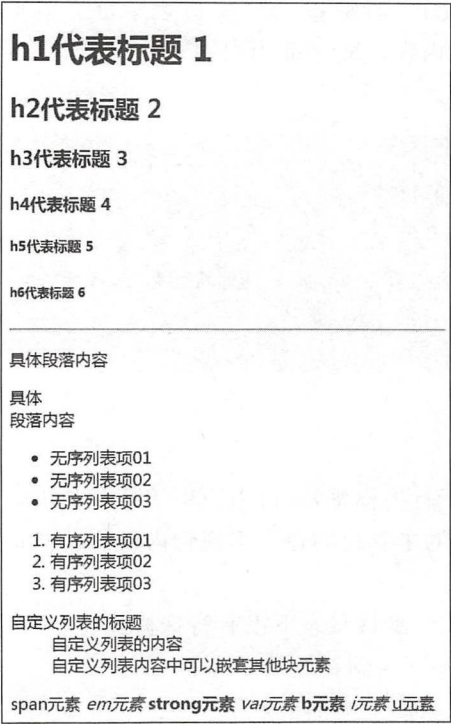


图 5.1 各类标签在浏览器中的显示效果





注意：该效果的 HTML 文件当中，并没有引入任何 CSS 文件，也没有书写任何 CSS 样式代码。虽然在默认状态下各个元素均具有一定的样式，但是在实际开发当中，会引入 CSS 重置文件，将默认样式“清除”之后，再拿来使用。



## 5.3 标签选择时的影响因素

掌握了在模块布局当中需要使用的各类标签之后，就是针对不同情况进行选择了。影响标签选择的因素主要有“标签默认样式”、“标签嵌套层数”、“标签的语义性(涉及 SEO)”、“标签的扩展性(涉及后台)”、“样式的可控性”以及“特殊功能需求(如：超链接)”等，如图 5.2 所示。接下来就一一剖析这几方面因素，并讲解涉及的相关知识。



图 5.2 标签选择时的各类影响因素



## 5.4 标签的默认显示样式

### 5.4.1 显示属性 display

属性功能：

规定元素展示的类型。

基本语法：

```
display: block;
```

代码解析：

设置元素以块元素的方式显示，元素会拥有块元素的相应特点。

属性值如表 5.1 所示。



表 5.1 display 的各种显示类型

值	描 述
none	此元素不会被显示
block	此元素将显示为块级元素,此元素前后会带有换行符
inline	此元素会被显示为内联元素,元素前后没有换行符
inline-block	此元素会被显示为行内块元素
list-item	此元素会作为列表显示(例如: <li>)
run-in	此元素会根据上下文作为块级元素或内联元素显示
table	此元素会作为块级表格来显示(例如: <table>),表格前后带有换行符
inline-table	此元素会作为内联表格来显示(例如: <table>),表格前后没有换行符
table-row-group	此元素会作为一个或多个行的分组来显示(例如: <tbody>)
table-header-group	此元素会作为一个或多个行的分组来显示(例如: <thead>)
table-footer-group	此元素会作为一个或多个行的分组来显示(例如: <tfoot>)
table-row	此元素会作为一个表格行显示(例如: <tr>)
table-column-group	此元素会作为一个或多个列的分组来显示(例如: <colgroup>)
table-column	此元素会作为一个单元格列显示(例如: <col>)
table-cell	此元素会作为一个表格单元格显示(例如: <td>和<th>)
table-caption	此元素会作为一个表格标题显示(例如: <caption>)
inherit	规定应该从父元素继承 display 属性的值

备注: 通过 display 属性改变标签展示类型,标签会拥有相应类型的标签特点。

5.4.2 根据标签默认 display 属性划分类别

不同的元素拥有不同的显示属性,最终呈现效果也并不相同,根据显示属性的不同,网页元素可以包含如下几种显示效果。

1. 块元素

块元素具备如下特性。

- (1) 默认独占父级的一行,不能够与其他元素处于同一行;
- (2) 能够设置宽高(换言之: 设置宽高有效);
- (3) 外边距设置生效。

常见的块元素包括:

- (1) 网页、框架等基本结构块: html、body、iframe。
- (2) 网页标题块: title。
- (3) 表单结构块: form、fieldset、legend。
- (4) 布局结构块: div。
- (5) 标题段落结构块: h1~h6、p。
- (6) 列表结构块: dl、dt、dd、ul、ol。
- (7) 结构装饰块: hr。





## 2. 行元素

行元素具备如下特性。

- (1) 默认由内容撑开宽高,与其他行元素能够处于同一行;
- (2) 不能够设置宽高(换言之:设置宽高无效);
- (3) 纵向外边距失效,横向外边距生效。

常见行元素包括:

- (1) 行内包含框: `span`。
- (2) 超链接: `a`。
- (3) 图像: `img`。
- (4) 各类文本修饰类标签: `abbr`、`acronym`、`b`、`bdo`、`big`、`cite`、`code`、`del`、`dfn`、`em`、`i`、`ins`、`kbd`、`q`、`s`、`samp`、`small`、`strike`、`strong`、`sub`、`sup`、`tt`、`u`、`var` 等。
- (5) 表单对象包含框: `select`、`button`、`label`、`textarea`。
- (6) 可执行对象包含框: `object`。

## 3. 第三类元素

其实除了块元素与行元素之外,还存在不少种类的元素,但由于这些种类的元素本身都不多,因此,将这些元素统一称为第三类元素。第三类元素可以具体划分为以下几种。

- |   |   |
|---|---|
| (1) <code>display: none;</code>               | ——头部隐藏元素: <code>head</code> 、 <code>link</code> 、 <code>meta</code> 、 <code>style</code> 、 <code>script</code> 等元素。 |
| (2) <code>display: list-item;</code>          | ——列表项元素: <code>li</code> 元素。  |
| (3) <code>display: table;</code>              | ——表格元素: <code>table</code> 元素。  |
| (4) <code>display: table-row;</code>          | ——表格行元素: <code>tr</code> 元素。  |
| (5) <code>display: table-header-group;</code> | ——表格头部分组显示: <code>thead</code> 元素。  |
| (6) <code>display: table-row-group;</code>    | ——表格行分组显示: <code>tbody</code> 元素。   |
| (7) <code>display: table-footer-group;</code> | ——表格底部分组显示: <code>tfoot</code> 元素。  |
| (8) <code>display: table-column;</code>       | ——表格列显示: <code>col</code> 元素。   |
| (9) <code>display: table-column-group;</code> | ——表格列分组显示: <code>colgroup</code> 元素。  |
| (10) <code>display: table-cell;</code>        | ——单元格显示: <code>td</code> 、 <code>th</code> 。  |
| (11) <code>display: table-caption;</code>     | ——表格标题显示: <code>caption</code> 。  |
| (12) <code>display: inline-block;</code>      | ——行内块状元素显示: <code>input</code> 、 <code>option</code> 、 <code>optgroup</code> 。                                      |

相关说明:

从本质上来说, `table` 表格与 `li` 列表项元素,都可以看作是块状元素,但是之所以不同分类,在于其各自包含一些独立性的特点。表格具有更加严格的组织性,表格元素之间的紧密协调性;列表拥有项目符号等一些其他特性。

### 5.4.3 显示样式影响的标签选用

各类元素都可以使用 `display` 属性进行元素类型的转换,例如,行和块元素可以通过 `display: block;` 和 `display: inline;` 命令相互转换。

在选用标签进行布局时,有时需要能够独自占据一行的标签,此时有两种方案,一种是直接选用块元素,另一种是选用行元素,再将行元素转换为块的样式来显示。很明显,直接

选用块元素会方便很多,这样就不需要增加 display 的代码。

## 5.4.4 display 的问题区

### 1. display 在开发中的实际用途

在真正网站的开发当中,使用较为频繁的 display 属性值,主要有 none 和 block。

对于超链接,有时为了提升用户体验,扩大超链接的区域大小,会针对 a 标签设置 display: block;

对于图像(img 标签),会设置浮动或 display: block;代码,将 img 标签下面的几个像素空隙去除掉。

当使用 JavaScript 制作一些网页特殊效果时,也会比较频繁地使用到 display: none;和 display: block。

有时会使用 display: inline-block;实现一个类似于浮动的效果。

其他的 display 类型,使用的会相对较少。

(关于超链接以及 img 的相关知识,在后面 2 章当中会详细讲解。)

### 2. display: none;与 visibility: hidden;

visibility 由于应用较少,在本书中没有详细讲解,它属于“显示类”属性,含义为“可视性”。

display: none;和 visibility: hidden;均能够实现元素的隐藏,所不同的是,设置 display: none;的元素并不会占据任何物理空间,而设置 visibility: hidden;的元素,虽然视觉上看不到了,但是依旧会占据物理空间。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - display:none 与 visibility:hidden 的区别</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    div {
      width: 200px;
      height: 50px;
      border: 1px solid black;
    }
    .dis {
      display: none;
    }
    .hide {
      visibility: hidden;
    }
  </style>
</head>
<body>
  <div class = "dis">第一个 div, 设置了 display: none</div>
  <div>第二个 div</div>
```



```
<div class="hide">第三个 div, 设置了 visibility: hidden</div>
<div>第四个 div</div>
</body>
</html>
```

所有的 div 元素均设置了 200 像素宽度, 50 像素高度以及 1 像素的实线黑色边框。针对第一个 div 元素设置了 `display: none;` 的属性, 针对第三个 div 元素设置了 `visibility: hidden;` 的属性。显示效果如下: 第一个 div 没有占据物理空间, 第二个 div 直接从页面的最左上方开始显示, 第三个 div 虽然看不到, 但是保留了原来的空间, 使得第四个 div 从纵向 100 像素的位置开始显示。

显示效果如图 5.3 所示。

### 3. 标签的展示类型能否修改

可以通过 `display` 属性修改标签的展示类型。但是, 在真正的开发当中, 并不推荐随意修改标签的展示类型, 在选择标签时, 应选择相应类型的合理标签。



图 5.3 `display: none;` 与 `visibility: hidden;`



## 5.5 标签的合理嵌套

### 5.5.1 标签嵌套基本规则

- (1) 块元素可以包含块元素和行元素。
- (2) 标题、段落类的块元素(包括 `h1~h6`, `dt`, `p`)不能够包含块元素。
- (3) 行元素可以包含行元素, 但不能够包含块元素。
- (4) `ins` 和 `del` 这两种行元素可以包含块元素。
- (5) `dl` 下只能直接包含 `dt` 和 `dd` 元素。
- (6) `ul`, `ol` 下只能直接包含 `li` 元素。
- (7) `table` 下只能直接包含 `caption`, `thead`, `tbody`, `tfoot`, `col`, `colgroup` 元素。
- (8) `thead`, `tbody`, `tfoot` 下直接包含 `tr`, `tr` 下可以是 `th` 或 `td`。通常 `th` 出现在 `thead` 的 `tr` 标签中。

备注: “直接包含”表示两种标签为父子级关系。

### 5.5.2 错误嵌套时的表现情况

#### 1. 标题类的块包含块元素

如果在标签选用时出现了错误, 使用了标题类块元素嵌套了块元素, 会不会出现问题呢?

## 代码实例 1:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 错误嵌套 1</title>
</head>
<body>
  <h1>
    <div>h1 中的第一个 div 元素</div>
  </h1>
</body>
</html>
```

在 h1 这种块元素当中嵌套了 div 元素,将代码在谷歌、IE、火狐等浏览器中运行之后,发现均能够正常显示,如图 5.4 所示。但是并不推荐这么做,因为它不符合嵌套规则。



图 5.4 h1 标签中嵌套 div 的显示效果

## 代码实例 2:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 错误嵌套 2</title>
</head>
<body>
  <h1>
    <h2>h1 中的 h2 元素</h2>
```



```

</h1>
</body>
</html>

```

在 h1 这种块元素当中嵌套了 h2 元素,将代码在谷歌、IE、火狐等浏览器中运行之后,发现除了 IE 浏览器之外,其他所有浏览器均不能够正常显示,h2 会自动被放置在 h1 的外面,如图 5.5 所示。

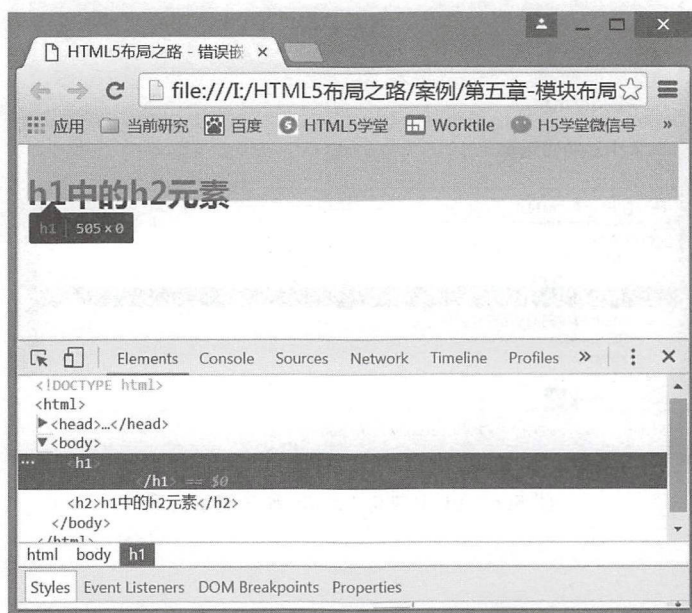


图 5.5 h1 标签中嵌套 h2 标签的显示效果

### 代码实例 3:

```

<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 错误嵌套 3</title>
</head>
<body>
  <h1>
    <h2>h1 中的 h2 元素</h2>
    <div>h1 中的 div 元素</div>
    <ul>
      <li>h1 中的 li 元素</li>
    </ul>
  </h1>
</body>
</html>

```

在 h1 元素中嵌套了一系列块元素,从上面的代码段 2 中能够知道,在谷歌、火狐等浏览器中,h2 元素会被浏览器“提取”到 h1 的外面,那么此时,div、ul 这些标签的位置在哪里呢?浏览器解析效果如图 5.6 所示。

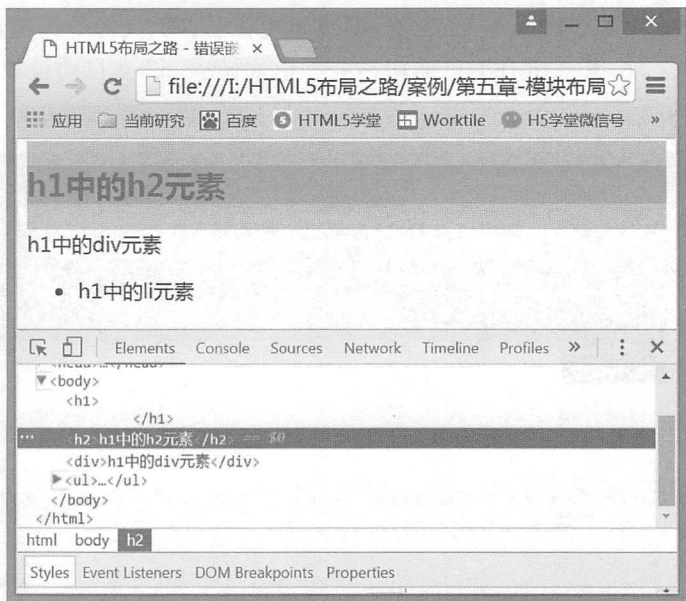


图 5.6 h1 中包含 h2、div 等各类块元素

#### 代码实例 4:

```
<!doctype html >
<html >
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - 特殊嵌套</title>
</head>
<body>
    <h1 >
        <a href = "" title = "">
            <h2 >h1 中非子元素的块元素</h2 >
        </a>
    </h1 >
</body>
</html>
```

这次没有把 h2 标签直接放置在 h1 当中,多嵌套了一层 a 标签,a 标签属于行元素,此时在各个浏览器中,均能够正常显示,如图 5.7 所示。

至此,做一个“标题类块元素”嵌套“块元素”的总结:

(1) 标题类的块元素在嵌套 div、ul、ol、p、dl 等块状元素时,浏览器解析时并不会出现问题,但是依旧不推荐这么操作。



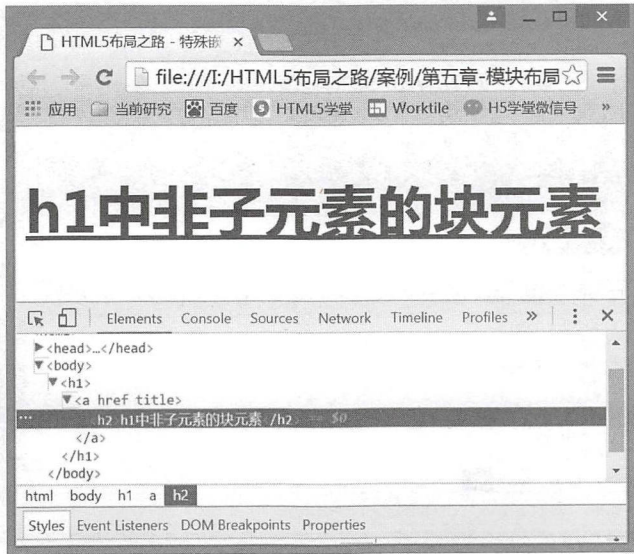


图 5.7 h1 中间接包含 h2 元素

(2) 标题类的块在嵌套 h2 等标题元素时,非 IE 浏览器解析会出现问题,h2 等子元素会自动被“提取”到标题父元素的外面。如果在当前子元素下还包括其他元素,会自动跟随该子元素,被“提取”到标题父元素的外面。

(3) 标题类的块能够包含非子元素的块状元素。但是,在真正开发过程中,只要标签选择合理,并不会出现这种情况。

2. 行元素包含块元素

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 行元素包含块元素</title>
</head>
<body>
  <span>
    <h2>span 行元素中的块元素 h2 </h2>
  </span>
</body>
</html>
```

在 span 这种行元素当中嵌套了 h2 这种块元素,在各个浏览器当中,均能够正常显示,如图 5.8 所示。

虽然能够正常显示,但是依旧不推荐使用行元素嵌套块元素(处于可触区的考虑,有时候会将 a 标签转换为块元素,在 a 标签中嵌套其他的块元素)。

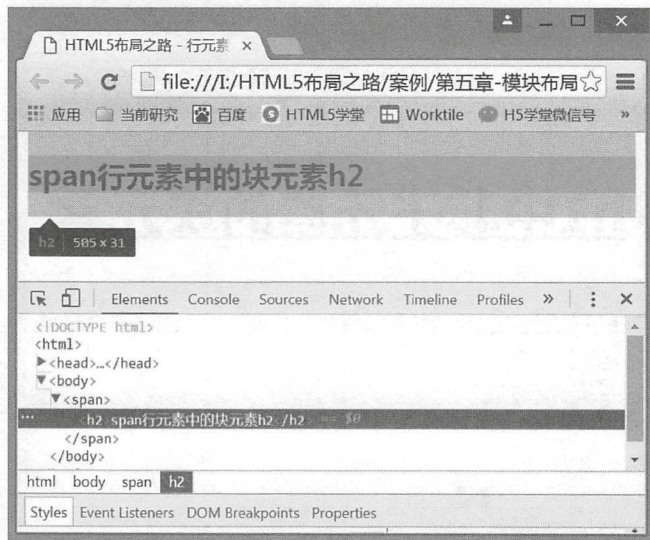


图 5.8 行元素中嵌套块元素



## 5.6 SEO 搜索引擎优化——标签语义性

### 5.6.1 为何要谈 SEO

#### 1. 为何开发一个网站

做个假设,如果你是一个企业老板,会不会考虑开发一个网站?或许你的网站能够辅助你的公司业务发展,或许你的网站就是你的公司业务,当然也有可能你的网站是拿来给自己员工内部使用的。

当前互联网当中的绝大多数网站的存在原因,都属于前两种,要么本身就是一个业务平台(如:京东、百度),要么就是和自己的公司、企业、单位机构业务相关,希望广大用户通过网站了解公司、公司业务等。

那么如何让广大用户了解到自己做的这个网站呢?可以借助电视广告、地铁广告,也可以到街上去发“小广告”,但是这些都是 20 世纪 90 年代人们信息的来源途径,而今人们更多的信息来源是互联网。

#### 2. 如何访问一个网站

例如希望寻找“HTML5 学堂”这个 HTML5 知识分享的网站,会通过什么途径来寻找呢?

##### 1) 通过 IP 地址

通过 IP,比如 192.XX.XX.XX。可是并不清楚每个网站的 IP 地址……

##### 2) 通过域名

HTML5 学堂的官网域名是 <http://www.h5course.com>。可是,这个域名会有多少人记得?即便记得,又有多少人愿意在地址栏当中输入它?

##### 3) 通过自己收藏的书签栏来实现快速访问



这是一种比较常用的方法,但是能够使用这种方式访问网站的用户,已不只一次进入过这个网站了。

#### 4) 通过百度等搜索引擎进行搜索

通过搜索引擎寻找资料如图 5.9 所示。



图 5.9 通过搜索引擎寻找资料

#### 5) 通过各大媒体的分享链接

通过各大媒体的文章链接跳转到网站中如图 5.10 所示。

大部分人寻找信息、网站的方式通常是后两种方式(搜索引擎和各大媒体网站的分享链接)。对于这两种来说,浏览者在访问时还有两个共同的特性,那就是“能看第一个就看第一个,能看第一页就看第一页”。

通常情况下,很少有浏览者会翻到搜索结果的第二页或第三页,几乎没有多少人会翻到搜索结果的第四页以及之后。此外,大部分浏览者都比较钟情于第一页的前几个搜索结果。

由于浏览者的浏览习惯,对于企业来说,网站在搜索引擎的排名就显得至关重要。让一个网站在搜索引擎中排名“靠前”,主要有两种方式方法,第一种是在用户使用搜索引擎进行某些相关词语搜索时,能够直接在搜索引擎中显示出这个网站,用户通过单击链接跳转到这个网站;另一种途径则是通过向搜索引擎付费的方式,使用户网站排名在前面。

需要注意的是,搜索引擎会先将推广链接(“付费”的企业网站)放在前面,而自然排名则放到推广链接的后面。

例如,搜索 HTML5 学堂,可以得到如下的搜索结果,其中第一篇文章就是推广链接,而从第二个链接才是自然排名。HTML5 学堂的官网,在这个页面中的自然排名在第二位。



图 5.10 通过各大媒体的文章链接跳转到网站当中

需要注意的是：推广链接，通常与真正的网站没有关系，只是通过付费而实现的排名，在各个搜索引擎当中，推广链接会加上“推广”的标识，如图 5.11 所示。



图 5.11 搜索引擎中的推广链接

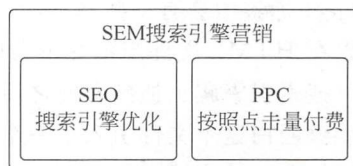


如此来看,网站的排名对于一个企业来说至关重要,那么如何让自己的排名提升上去呢?除了可以花钱做推广和宣传之外,还可以努力地提高自然排名,当然也可以两者相结合。关于网站的排名优化,就是接下来要讲到的 SEO。

### 5.6.2 SEO 是什么

SEO 是 Search Engine Optimization(搜索引擎优化)的缩写,是指为了从搜索引擎中获得更多的免费流量,从网站结构、内容建设方案、用户互动传播、页面等角度进行合理规划,使网站更适合搜索引擎的检索原则的行为。

SEO 与 PPC(按照点击量付费),共同组成了 SEM(Search Engine Marketing, 搜索引擎营销),如图 5.12 所示。



### 5.6.3 搜索爬虫工作原理

搜索引擎构建一个调度程序,来调度“蜘蛛”的工作,让“蜘蛛”去和服务器建立连接下载网页,计算的过程都是通过调度来计算的,“蜘蛛”只是负责下载网页。

通过“蜘蛛”下载回来的网页放到补充数据区,通过各种程序计算过后才放到检索区,之后形成稳定的排名,当网页存在于补充数据区时,是不稳定的,因为它随时都有可能在计算的过程中被 PASS 掉,而检索区的数据排名是相对比较稳定的。

由于数据来源于数据库(前面提到的检索区和补充数据区域),而不是当时的链接抓取,这也就很好地解释了“快照”的含义,这种处理方法能够很好地提升页面打开速度,如图 5.13 所示。

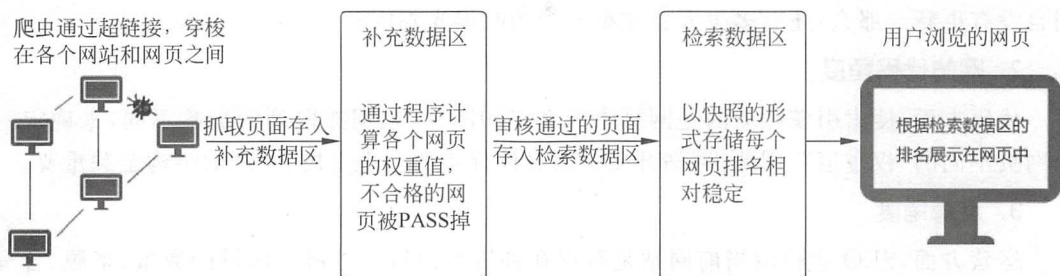


图 5.13 搜索引擎爬虫工作原理

### 5.6.4 爬虫抓取的是什么

可用两个词来概括爬虫抓取的内容,那就是图片、文字。

在 2005 年之前,前端开发需要负责“设计和制作”两种工作,因此也可以被称为“网页设计与制作”,开发的工具被称为“网页三剑客”,分别是 Photoshop、Flash 和 Dreamweaver。

在那个时代,负责网页行为的 JavaScript 还是一个“跳梁小丑”的角色,通常就是制作一个当前时间、文字滚动之类的小效果。对于网页中比较大型的动态效果,很多都是由 Flash 制作出来的,甚至整个网站都采用的是 Flash。

查阅很多网页设计与制作的书籍时,常常会在附录中找到不少“优秀网站”的链接地址,打开附录中的各个网站,看到的都是非常酷炫的效果。但是,这些看上去高大上的网站,存在着一个共同的问题——使用百度等搜索引擎进行查找时,无法搜索到当前网站,即便使用的是与这个网站内容非常贴切的词语,也无济于事。

造成这个问题的根源就是上面提到的,搜索引擎的爬虫,只抓取图片和文字,当使用Flash进行网站制作时,在HTML文件的<body>当中,会有且仅有一个<object>标签(object标签中会有一些基本的设置),而由于所有的文本和图片都是封装在Flash当中的,所以在HTML文件当中并不会以任何文字或图片的形式呈现出来。

搜索引擎爬虫抓到页面之后一看,没有关于这个网站的什么信息,仅有一个object,爬虫能够明白这个文件引入了一个“对象”,除此之外,爬虫一无所知。在这样的情况下,无法得到排名也就在情理之中了。

## 5.6.5 什么样的网站才能够被快速收录

### 1. 网站内容角度

内容方面,搜索引擎要求网站要“更新频繁并且原创”。

搜索引擎爬虫对一个网站访问的频率,直接与网站的更新频率和原创程度挂钩。更新的相对比较频繁时,搜索引擎爬虫会定期来访,且时间间隔会比较短;对于更新不频繁的网站,搜索引擎爬虫光临的时间也就相对比较长。

这种访问的特点与人的行为特点也比较类似,下面举一个生活上的例子来理解一下。

假设我们是一部动漫的爱好者,这部动漫每周更新一次,于是我们会每隔一周去看一下更新的动漫;但是突然有一周,它没有更新了,这时发现没有更新,我们会想:什么情况?为什么没有更新?在第二周,可能还会过来看一次。可是,第二周也没有更新,第三周、第四周也没有更新。那么,还有多少人会在第五周的时候光临呢?

### 2. 网站代码角度

代码方面,搜索引擎抓取的是网页中的文字和图片,根据抓取到的标签不同,来确定一个网页不同的“权重值”,因此,网站开发中语义性标签的合理选用、合理的结构至关重要。

### 3. 运营角度

运营方面,SEO会检测当前网站是否存在外链和内链。外链和内链的数量、来源、合理性都会有利于SEO。

#### 1) 外链

外链指的是其他网站中能够跳转到当前网站的链接。这些工作需要公司的推广人员进行,运营百度贴吧、发帖、发评论、运营博客、在各个媒体平台发表相关的文章,并将链接指向需要优化的网站。

对于一个网站来说,当它的链接出现在各个其他知名网站时,也就意味着这个网站很有名气,SEO就会认为这个网站权重级别更高。

#### 2) 内链

内链指的是当前网站自身当中,某一个页面跳转到当前网站其他页面的链接。内链的操作,多数是由公司的运维进行的。



当一个网站内部的链接合理,一个网站的各个网页就不再是独立的,而是组成了一个复杂的关系网络,搜索引擎爬虫可以顺着一个链接,查找到更多的网页,也是有利于 SEO 的。

### 5.6.6 针对 SEO,前端开发要注意什么

#### 1. 代码规范

##### 1) 扩展性良好

SEO 并不能够像人一样,通过查看代码查看效果从而判断扩展性。但是 SEO 能够通过审查相关因素来审查代码扩展性。这个相关因素就是——代码大小与加载速度。通常情况下,扩展性强的代码,就不会存在重复性的功能代码,而此时,文件大小相对较小,服务器加载速度就会比较快。

##### 2) 格式规范

此前提到过,每一个级别的标签都需要添加一次 Tab 键,进行缩进。除了让我们自己能够清晰查看代码之外,还有一点在于:如果缩进、格式混乱、各种空行、标签书写位置有误等,都会有可能让爬虫认为这段代码是机器生成的,从而影响 SEO 排名。

##### 3) 合理嵌套

如果标签没有合理嵌套,出现了最典型的错误,也会影响 SEO 排名。

#### 2. 标签的语义性

爬虫抓取的是网页文档中的文字和图片,那么爬虫如何知道某段文字在网页中充当的是什么角色呢?

此时,爬虫会读取包含文字的各种标签,由于每种标签都带有一定的含义,因此,不同标签中的不同文字在网站中的“地位”也就有所不同。

“地位”比较重要的标签如下。

(1) 整个网页的标题: `<title></title>`。

(2) 网页的描述信息: 元信息 meta 中的描述信息 description。

(3) 一级标题~三级标题: `<h1></h1>`; `<h2></h2>`; `<h3></h3>`。

(4) 图片的替代性文本: img 标签的 alt 属性。

(5) a 标签的提示性文本: a 标签的 title 属性。

(6) 列表、表格等元素的合理运用: ol、ul、dl、table 等。

备注: 四级标题的重要度较低,并不是很常用。

常用的各类标签以及含义,请详见此前的 2.3 节。

#### 3. 重要的部分尽量为纯静态

有时候,为了追求网页的特殊呈现效果,会使用 JavaScript 的 AJAX 技术,实现网页中的部分数据甚至全部数据的动态加载。但是,从爬虫的搜索角度来说,并不能够检查到 JavaScript 控制的 DOM 部分(所谓的 DOM,可以理解为网页中的各个标签和文字内容),因此建议,对于一些比较重要的部分,不要使用 JavaScript 进行动态加载,而是采用生成好的静态标签。

此外,还要注意不要使用图片去替换重要的文本内容。例如,当前存在一个 LOGO, LOGO 是由一段文本和一个特殊图像组成的,此时,一种操作方法是,将标题和图像作为一

个整体,截取成一张图片,之后为截取后的这个图片添加链接;另一种操作方法是,单独截取这张图像,标题依旧使用 HTML 的相关标签进行书写,之后通过布局的手段将这个图像和标题恢复为 LOGO 中的效果。

这两种操作方法在最终显示效果上相同,从操作层面上前者要更简单,但是,前者在 SEO 方面要比后者差很多。

#### 4. 锚文本

锚文本,指的是<a></a>标签中的具体文字,锚文本是由页面需要优化的关键字来决定的,如图 5.14 所示。

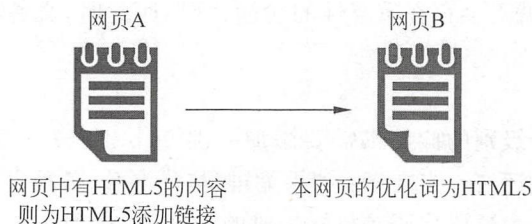


图 5.14 锚文本

希望用户搜索什么词语能够找到一个网页,那么这个词就是这个网页“需要优化的关键词”。对于每一个网页,由于内容的不同,需要优化的关键词也有所不同。

例如,H 网站中有很多网页,其中,A 网页的具体内容是关于 SEO 方面的具体介绍,A 网页的所有者希望:各个用户通过在搜索引擎搜索“SEO”能够找到 A 网页。于是,A 网页需要优化的关键词就是“SEO”。

当在 H 网站中的其他网页中出现了“SEO”的词语时,就可以为这个词添加链接,链接到 H 网站的 A 网页当中。

#### 5.6.7 SEO 中表示强调的标签

在 SEO 当中,有两种行内标签表示强调语气,分别是 strong 和 em。

strong、b、em、i 这 4 种行标签比较相似,在此给出它们彼此之间的区别。

(1) b、i 属于修饰类标签,也叫物理标签;strong、em 属于内容类标签,也叫逻辑标签;物理标签是说告诉浏览器在哪里加粗了,没有别的作用,而逻辑标签是强调语气,它强调语气是通过文本加粗来体现的,也可以通过别的样式来强调语气。

(2) b、strong 标签表现为加粗样式;em、i 表现为倾斜样式。

(3) strong、em 表强调;strong 比 em 语气更强烈。

(4) 在搜索引擎优化中 strong 和 em 比 b 和 i 重要得多。

#### 5.6.8 关于 SEO 的问题区

(1) 是不是所有的媒体的外链都有效?

并非所有媒体上的链接都称为外链。

对于 QQ 空间、微信等上的各种网站外部链接,并不会对网站的 SEO 有帮助,但是的确可以为网站产生“用户流量”。究其原因在于,这些媒体本身是独立的一部分,可以简单理解



成：它们是一个独立的盒子，封闭起来，不对外打开。既然不对外打开，在搜索引擎中，自然就不能够搜索到 QQ 空间的文章了，这些媒体平台也不能够对外部网站的排名有所帮助了。

(2) 通常从网站开发完成，到被搜索引擎收录，再到排名第一，需要多久？

通常，只要网站保持正常更新，配合适当的营销手段，大概一个月，就能够被搜索引擎收录。如果想将网站做到搜索引擎排名第一位，除了和网站本身相关之外，还与搜索词的热度相关，搜索词热度越热，越难提升排名，所花的时间也越长。少则两三个月，多则几年。

(3) 博客类网站的搜索机制有何不同？

很多写博客的人能够发现，即便自己的博客没有太多的更新文章，自己的文章仍然可能被各个搜索引擎所收录。有时，虽然自己很久没有书写文章，但是搜索爬虫仍然会光临。

这种现象，主要与博客的搜索机制相关。可以简单地认为，在博客类的网站当中，所有的文章发布之后会定期地在一个数据文件中更新，搜索爬虫的光临会对所有的数据读取，从而使得数据能够被搜索引擎收录。

此时，只要保证这个数据文件在更新，那么爬虫就会定期光临并读取所有数据。

(4) 白帽优化与黑帽优化。

生活中有句俗语“无论白猫黑猫，抓住耗子都是好猫”。但是，在优化层面，黑帽优化是开发者不应该去触及的。

白帽优化，是使用符合主流搜索引擎发行方针规定的 SEO 优化方法。它与黑帽优化相反，白帽优化一直被业内认为是最佳的优化手法，它是在避免一切风险的情况下进行操作，同时也避免了与搜索引擎发行方针发生任何的冲突，它也是 SEO 从业者的最高职业道德标准。

白帽优化关注长远利益，需要优化的时间比较长久。坚持不使用作弊手段，一段时间之后，不出意外的话，网站就应该可以得到好的流量和排名，也就有了盈利点，而且后续对搜索引擎的依赖度更小。从长远的利益发展建议还是用白帽手法。

黑帽优化，是一种 SEO 作弊手法，当前各个浏览器对其会严重查处，一旦发现，网站就不得翻身，主要包括博客作弊、群发软件作弊、挂黑链、链接养殖场等手法。

(5) 建议前端了解的几个 SEO 名词和扩展性知识。

① PPC: Pay Per Click, 即“按效果付费”模式，也被称作 PFP(Pay-For-Performance, 按业绩付费)，是目前广告市场中最具有竞争力的广告计费模式之一。

一个搜索词的价钱 = 起价 + 点击量 × 每次点击的价格。越是著名的搜索引擎，起价就会越高，最高可达数万甚至数十万。而每次点击的价格在 0.30 元左右。提供点击量付费的网站非常多，主要有各大门户网站（如搜狐、新浪）、搜索引擎（如 Google、百度和腾讯搜搜），以及其他浏览量较大的网站，比如提供软件下载的华军等。

② PR 值：PR 值即 PageRank，即网页的级别技术。取自 Google 的创始人 Larry Page，它是 Google 排名运算法则（排名公式）的一部分，用来标识网页的等级/重要性。级别从 0 到 10 级，10 级为满分。PR 值越高说明该网页越受欢迎（越重要）。例如，一个 PR 值为 1 的网站表明这个网站不太具有流行度，而 PR 值为 7 到 10 则表明这个网站非常受欢迎（或者说极其重要）。

一个网站的外部链接数越多其 PR 值就越高；外部链接站点的级别越高（如 Macromedia 的网站链接了你的网站），网站的 PR 值就越高。

③ PV 与 UV：PV 指的是浏览量；UV 指的是用户量。需要注意的是，如果一个用户浏览了当前网站中的多个网页，他每浏览一个网页，PV 值都会+1，但是 UV 值不会发生变化。

④ 跳出率：跳出率是指用户到达网站上并在网站上仅浏览了一个页面就离开的访问次数与所有访问次数的百分比。

如果已经吸引用户来到了自己的网站，那么应该要满足用户的期望与需求。不同网站对跳出率的要求也有所不同，有些网站跳出率越高越好，例如，网址导航类的网站；而有的网站则是跳出率越低越好，例如，京东淘宝等电商平台。

⑤ SEO 综合查询网址：<http://seo.chinaz.com/>。



## 5.7 嵌套层数与深度

在选用标签时，还需要考虑嵌套的层数，嵌套的层数多了，一方面会让整体代码增多，另一方面 SEO 并不能够抓取非常深的结构代码。在选择标签时，要尽量减少结构的嵌套。



## 5.8 标签选择实战(1)——确定标签

### 5.8.1 功能需求

图 5.15 是比较典型的文章列表页中的一部分。左侧图片是文章的预览图，右侧是文章的标题、发布时间、文章类别、描述信息。



#### 经典的斐波那契数列与arguments.callee

2016-08-15      文章类别：JavaScript

HTML5学堂：经典的斐波那契数列与arguments.callee。提到斐波那契数列，很多人还不是太清楚，但是如果提到兔子的繁殖这个经典题目，相信学过计算机语言的人们会立刻感觉亲切



#### WEBP格式的图片

2016-07-30      文章类别：JavaScript

HTML5学堂：谷歌于2010年推出的新一代图片格式——WEBP，随着移动互联网的发展，WEBP格式在2015年逐渐地开始被大公司部分采用。



#### 使用Git多人协作，完成项目开发

2016-07-10      文章类别：工具类文件

HTML5学堂：关于Git的知识，我们共分成了四个大步骤进行讲解，之前我们提到了Git的安装与配置、Git在本地的使用方法，如何创建Git本地仓库与服务器端仓库的关系。

图 5.15 标签选择实战功能需求图



## 5.8.2 提出实现方案

### 1. 方案1——使用 ul 无序列表

代码实例：

```
<ul>
  <li>
    <div><img src="" alt="" title=""></div>
    <div>
      <h2>文章大标题</h2>
      <div><span>年月日</span><span>文章分类</span></div>
      <p>文章描述信息</p>
    </div>
  </li>
</ul>
```

### 2. 方案2——使用 dl 自定义列表

代码实例：

```
<dl>
  <dt><img src="" alt="" title=""></dt>
  <dd>
    <h2>文章大标题</h2>
    <div><span>年月日</span><span>文章分类</span></div>
    <p>文章描述信息</p>
  </dd>
</dl>
```

### 3. 方案3——使用内部结构不同的 dl 自定义列表

代码实例：

```
<dl>
  <dt><img src="" alt="" title=""></dt>
  <dd>
    <h2>文章大标题</h2>
    <h3><span>年月日</span><span>文章分类</span></h3>
    <div>文章描述信息</div>
  </dd>
</dl>
```

### 4. 方案4——大量应用 div

代码实例：

```
<div>
  <div><img src="" alt="" title=""></div>
  <div>
```



```
<div>文章大标题</div>
<div>< span>年月日</span>< span>文章分类</span></div>
<div>文章描述信息</div>
</div>
</div>
```

5. 方案5——使用 table 表格

代码实例：

```
<table>
  <tbody>
    <tr>
      <td>< img src = "" alt = "" title = ""></td>
      <td>
        <h2>文章大标题</h2>
        <div>< span>年月日</span>< span>文章分类</span></div>
        <p>文章描述信息</p>
      </td>
    </tr>
  </tbody>
</table>
```

5.8.3 标签选择思路分析

图 5.16 是一系列文章的简介,它和其他的文章简介共同构成了一个文章列表,包含多篇文章的列表,能够想到的就是 ul、ol 和 dl 三种列表类标签。

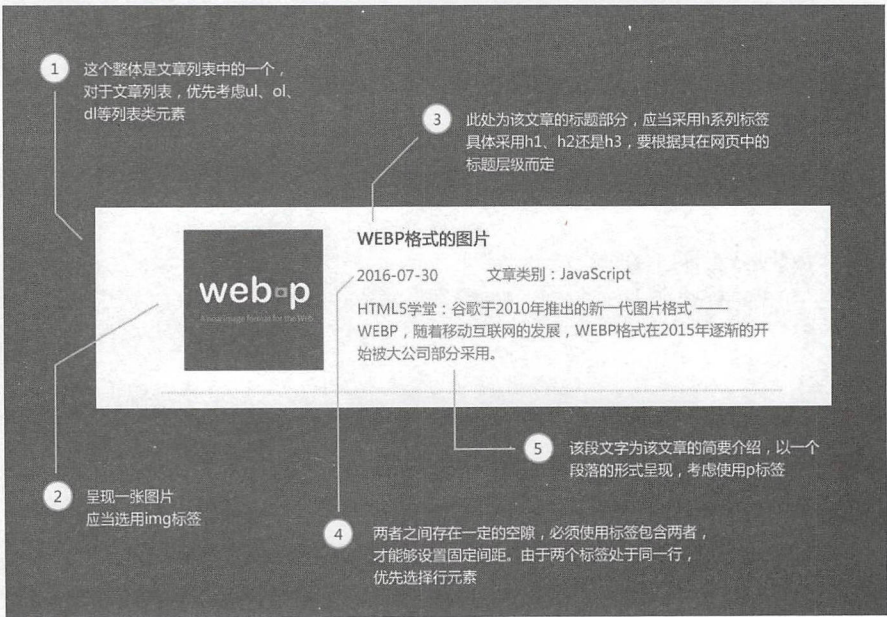


图 5.16 标签选择思路分析





在这个文章简介中,左侧的图和右侧的标题、内容应当是一个整体,那么在搭建结构时就不能够将两者彻底分开,要有一定的联系,例如,同属于某一个父级元素。

左侧图像部分需要使用到 `img` 标签,右侧的文章标题部分考虑使用 `h1~h3` 中的某一种,段落类的介绍使用 `p` 标签,对于时间和类别,两者样式相同、存在空隙,因此可以采用同一种行内标签来进行处理。

5.8.4 实现方案的对比分析

经过上面的标签选择的分析之后,从嵌套合理、层数深度、标签语义性等角度将上面的5种方案进行一下对比,如表 5.2 所示。

表 5.2 标签选择案例实现方案对比

方案	具体方案	嵌套情况	标 签 语 义	评价	备 注
1	ul 无序列表	合理嵌套	合理	较好	ol 和 ul 类似
2	dl 自定义列表	合理嵌套	合理	较好	使用 div 标签包含发布时间和文章分类
3	dl 自定义列表	合理嵌套	在 h3 部分有误	一般	使用 h3 标签包含发布时间和文章分类
4	大量应用 div	合理嵌套	没有语义性	差	
5	table 表格	层数太多	语义性偏差	差	table 标签的语义性为表格

能够看出,方案 1 和方案 2,都能比较合理地传达出语义性;在嵌套方面,也都遵循基本的嵌套规则,层数基本合理。那么,方案 1 和方案 2 哪个会更好呢? 决定最终选择的还有其他因素,一起继续往下看。



5.9 样式的可控性

5.9.1 原有选择器对样式的控制问题

此前已经讲解过了三种基本选择器,分别是 ID 选择器、类名选择器、标签名选择器。

这三种基本选择器虽然能够实现样式的控制,但是在运用过程中,却出现了一些值得思考的问题。

稍微完善一些上面的实例代码。

代码实例:

```
<dl class = "list - item">
  <dt class = "list - picbox"><img class = "list - pic" src = "" alt = "" title = ""></dt>
  <dd class = "list - con">
    <h2 class = "list - tit">文章大标题</h2>
    <div class = "list - inf"><span class = "update">年月日</span><span class = "kinds">
文字分类</span></div>
    <p class = "list - des">文章描述信息</p>
  </dd>
</dl>
```

为了控制每个标签的样式,需要为每个标签设置类名,刚刚看上去比较简洁的代码,一



下子变得冗长复杂起来,难道就没有别的办法让我们稍微轻松一点儿吗?

为了便于样式的控制,少起一些类名,减少代码量,也减少工程师起名字的烦恼,CSS为我们提供了“加强版”的选择器,先来了解一下这些加强版的选择器,再回到这个实例。

### 5.9.2 加强版选择器——后代选择器

基本语法:

```
选择器名 1 选择器名 2 ... 选择器名 n{
    属性名: 属性值;
    属性名: 属性值;
    :
}
```

选择器代码:

```
#wrap p {
    border: 1px solid red;
}
```

代码实例:

```
<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - 后代选择器</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        #wrap p {
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <div id = "wrap">
        <p>段落 1 </p>
        <p>段落 2 </p>
        <p>段落 3 </p>
        <div>
            <p>div 中的 div 中的段落</p>
        </div>
    </div>
</body>
</html>
```

代码解析:

#wrap p {border: 1px solid red;}的意思是为 id 为 wrap 中里面所有的 p 元素添加 1 像素实线红色边框的样式。需要注意的是不单单是子代受影响,所有的后代均会受到影响。





后代选择器就是使用多个选择器的组合来找到要控制的标签,不同的选择器之间使用空格分隔。

在上面的实例当中采用的是两种选择器组合而成的后代选择器,需要注意的是,后代选择器的组成并不局限于两个选择器,可以是多个,如: `.wrap div p{}`; `.wrap .con dl dt a{}` 此类的多个选择器组成的“后代选择器”。

注意:

由于后代选择器选择到的是所有符合条件的后代,因此要防止一些元素受到影响,在选择标签的时候要额外慎重。

### 5.9.3 加强版选择器——子代选择器

基本语法:

```
选择器名 1 > 选择器名 2 > ... > 选择器名 n {  
    属性名: 属性值;  
    属性名: 属性值;  
    ...  
}
```

选择器代码:

```
#wrap > p {  
    border: 1px solid red;  
}
```

代码实例:

```
<!doctype html>  
<html>  
<head>  
    <meta charset = "UTF-8">  
    <title> HTML5 布局之路 - 子代选择器</title>  
    <link rel = "stylesheet" href = "../css/reset.css">  
    <style>  
        #wrap > p {  
            border: 1px solid red;  
        }  
    </style>  
</head>  
<body>  
    <div id = "wrap">  
        <p>段落 1</p>  
        <p>段落 2</p>  
        <p>段落 3</p>  
    </div>
```



```

        <p>div 中的 div 中的段落</p>
    </div>
</div>
</body>
</html>

```

**代码解析：**

#wrap > p {border: 1px solid red;} 中的意思是 id 为 wrap 中里面的子代 p 元素添加 1 像素实线红色边框的样式。在显示效果当中,段落 1~段落 3 的 p 标签被赋予了这个样式,但是“div 中的 div 中的段落”(第 4 个 p 标签)并没有被设置 1 像素红色边框。

子代选择器也是使用多个选择器的组合来找到要控制的标签,不同的选择器之间使用大于号“>”分隔。整体的原理与后代选择器类似,所不同的是,子代选择器仅仅选择到的是一代,而非所有后代。

关于兼容:子代选择器在 IE6 中存在兼容问题,不能够使用;在其他各个浏览器中均能够正常使用。

**5.9.4 加强版选择器——群组选择器****基本语法：**

```

选择器名, 选择器 2, ..., 选择器 n{
    属性名: 属性值;
    属性名: 属性值;
    ...
}

```

**选择器代码：**

```

#wrap, .con p {
    border: 1px solid red;
}

```

**代码实例：**

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF-8">
    <title>HTML5 布局之路 - 群组选择器</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        #wrap, .con p {
            border: 1px solid red;
        }
    </style>

```





```

</head>
<body>
  <div id="wrap">
    <p>段落 1</p>
    <p>段落 2</p>
    <p>段落 3</p>
    <div class="con">
      <p>div 中的 div 中的段落</p>
    </div>
  </div>
</body>
</html>

```

#### 代码解析:

# wrap, .con p {border: 1px solid red;}的意思是 id 名为 wrap 的元素添加 1 像素实线红色边框的样式; 为类名为 con 的元素中的所有 p 元素添加 1 像素实线红色边框的样式。

群组选择器是为“多个不同的选择器”设置“相同的样式”,不同的选择器之间使用逗号进行分隔。

在一个网站当中,通常在各个页面中的存在着类似的部分,或有些模块或元素的样式是一样的,开发工程师虽然可以分别针对这些元素进行样式的书写,但是这样的操作会使代码修改变得很不方便,代码量也会变得更多,此时可以使用群组选择器来解决这个问题。使用群组选择器,统一设置通用样式时,既能够减少页面中的代码量,也方便代码的后期修改与维护。

### 5.9.5 加强版选择器优先级算法

在之前,讲解过 ID 选择器、类名选择器以及标签名选择器的优先级,那么对于后代、子代、群组选择器,优先级是如何计算的呢?

- (1) 对于后代、子代选择器,其优先级是进行叠加运算(叠加但不进位);
- (2) 对于群组选择器,每个逗号分隔不同的选择器,不同的选择器各自计算各自的优先级。

#### 代码实例:

```

# wrap, .con p {
  border: 1px solid red;
}

```

#### 代码解析:

# wrap, .con p 是一个群组选择器,包含两种选择器,一种是“# wrap”,另一种是“.con p”;这两种选择器的优先级分别进行计算。对于“id 名为 wrap 的元素”,该选择器的优先级是“0100”;对于“类名为 con 元素中的 p 元素”,该选择器“.con p”是由一个类名选择器和一个标签选择器组成,类名选择器的优先级是“0010”,标签名选择器的优先级是“0001”,因此这个后代选择器的优先级是“0011”。

代码实例：

```
.con .a1 .a2 .a3 .a4 .a5 .a6 .a7 .a8 .a9 .a10 p {
    border: 1px solid red;
}
#wrap {
    border: 1px solid blue;
}
```

如上代码当中,第一种选择器是一个后代选择器,这个后代选择器由 11 个类名选择器和 1 个标签名选择器共同组成。第二种选择器是一个 id 选择器。假设这两种选择器都选择到同一个标签,那么此时,这个后代选择器优先级是多少,边框到底是红色还是蓝色?

代码解析：

第一种(后代)选择器组成: 11 个类名选择器,优先级是 00110,1 个标签名选择器,优先级是 0 0 0 1。叠加起来是 0 0 11 1,而不是 0 1 1 1。

第二种(ID)选择器: 优先级是 0 1 0 0。

比较这两种选择器,依旧是 ID 选择器的优先级更高,因此,如果这两种选择器都选择到同一个标签,这个标签的边框颜色是蓝色。

**注意:** 在真正的开发当中,嵌套层数并不会这么多,开发工程师书写选择器时,也不会出现这种“十几个类名组合成后代选择器”的书写方法。在这里只是希望通过这个实例,让大家了解,选择器的优先级不能够“进位”。

## 5.9.6 关于样式可控性的问题区

### 1. 还有哪些种类的选择器可用来控制样式

在 CSS2.0 当中,除了为开发者提供了三种基本选择器、三种加强版选择器之外,还存在“通配符选择器”、“毗邻选择器”、“伪类选择器”和“属性选择器”。

关于通配符选择器和毗邻选择器,在开发中使用的并不是很多,一会儿做一下简要介绍。

伪类选择器,通常应用于 a 标签,在 6.1 节当中,会详细讲解伪类选择器。

属性选择器,比较频繁地应用于表单元素当中,在第 11 章“表单”当中,会进行详细的讲解。

### 2. 通配符选择器

CSS 中,使用“\*”表示通配符选择器。使用通配符,能够选择到所有的标签。

选择器代码：

```
* {
    border: 1px solid red;
}
```



**代码解析:**

该代码表示,为网页中所有标签添加 1 像素的红色实线边框。

通配符选择器的选择范围比标签名选择器还要广,因此在开发中并没有得到使用。

**3. 毗邻选择器**

毗邻选择器,也称为相邻选择器。使用“+”号连接两个选择器。IE6 浏览器不支持。

**选择器代码:**

```
.con + p {
    border: 1px solid red;
}
```

**代码实例:**

```
<!doctype html>
<html>
<head>
    <meta charset = "UTF-8">
    <title>HTML5 布局之路 - 毗邻选择器</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        .con + p {
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <div class = "con">类名为 con 的 div 元素</div>
    <p>在类名为 con 的 div 元素 后面的 第一个 p 元素</p>
    <p>在类名为 con 的 div 元素 后面的 第二个 p 元素</p>
    <div class = "con">类名为 con 的第二个 div 元素</div>
    <h1>在类名为 con 后面,但是是 h1 元素</h1>
    <div class = "con">
        <p>存在于类名为 con 元素内部的子元素,p 元素</p>
    </div>
</body>
</html>
```

**代码解析:**

.con + p 选择器表示,选择类名为 con 的元素后面的 p 元素,且要求 p 元素必须是 div 元素的兄弟级元素。

显示效果如图 5.17 所示。

**4. div .wrap 与 div.wrap 是同一种选择器吗**

div .wrap{}: 在该选择器当中,div 与 .wrap 之间有一个空格,因此表示后代选择器,选中的是 div 元素下的类名为 wrap 的元素。

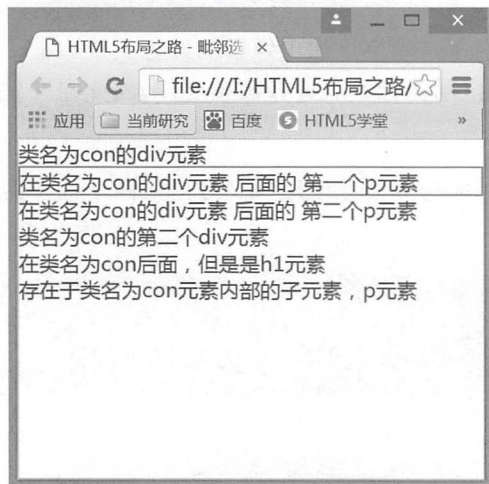


图 5.17 毗邻选择器案例效果图

`div.wrap {}`: 在该选择器当中, `div` 与 `.wrap` 之间没有任何内容, 直接连接在一起, 此时并非表示后代选择器, 而是表示符合某种条件的某种选择器, 此处选择器选中的是类名为 `div` 的 `div`。

其他代码实例:

`div.con # box { }` 表示“类名为 `con`”且“id 名为 `box`”的 `div` 元素。



## 5.10 标签选择实战(2)——样式控制

### 5.10.1 方案 1——使用 `ul` 无序列表

有了后代等加强版的选择器, 样式的控制就变得简单了很多, 再也不需要为每个标签起一个单独的类名了。原来的代码可以变成如下状态。

代码实例:

```
<ul class="arc-list">
  <li>
    <div class="list-picbox"><img src="" alt="" title=""></div>
    <div class="list-con">
      <h2>文章大标题</h2>
      <div><span>年月日</span><span>文字分类</span></div>
      <p>文章描述信息</p>
    </div>
  </li>
</ul>
```

由于要区分开 `li` 元素中的两种 `div` 元素, 需要为这两种 `div` 元素设置类名。对于各个标签以及类名控制, 如表 5.3 所示。



表 5.3 方案 1 的选择器选用分析

要设置样式的元素	对应选择器	要设置样式的元素	对应选择器
ul 最外层包含框	. arc-list	img 图像	. list-picbox img
li 每一个列表	. arc-list li	h2 文章大标题	. list-con h2
包裹 img 的 div 元素	. list-picbox	包含日期、文字的 div	. list-con div
包裹内容的 div 元素	. list-con	span	. list-con span
p 文章描述信息	. list-con p		

**备注：**对于年月日、文字分类的 span 元素，由于显示效果一致且两者之间存在一定的距离，因此选用同一种标签进行控制，在对应选择器当中使用了 . list-con span，如果使用 “. list-con div span”或“. arc-list li . list-con div span”选择器都是可以的，“. list-con div span”选择器表示的是选择到类名为 list-con 元素中的 div 元素里的 span 元素，针对 span 元素进行了进一步的限制，而“. arc-list li . list-con div span”选择器则是一层一层书写下来。

在实际开发当中，要基于实际的标签结构和后代选择器的基本原理，考虑具体选用哪种选择器，使用什么样的选择器名称。在该实例中，列表结构内部并不存在其他的 span 元素，因此，可以直接省略掉很多其他内容。采用最简单的书写方法，即“. list-con span”或“. arc-list span”。

### 5.10.2 方案 2——使用 dl 自定义列表

代码实例：

```
<div class="arc-list">
  <dl>
    <dt><img src="" alt="" title=""></dt>
    <dd>
      <h2>文章大标题</h2>
      <div><span>年月日</span><span>文字分类</span></div>
      <p>文章描述信息</p>
    </dd>
  </dl>
</div>
```

对于各个标签以及类名控制，如表 5.4 所示。

表 5.4 方案 2 的选择器选用分析

要设置样式的元素	对应选择器	要设置样式的元素	对应选择器
div 最外层包含框	. arc-list	img 图像	. arc-list img
dl 每一个列表	. arc-list dl	h2 文章大标题	. arc-list h2
包裹 img 的 dt 元素	. arc-list dt	包含日期、文字的 div	. arc-list div
包裹内容的 dd 元素	. arc-list dd	span	. arc-list span
p 文章描述信息	. arc-list p		

对比两种方案发现，由于自定义列表 dl 当中，会细化分为自定义列表标题 dt 和自定义列表内容 dd 两种，因此，能够代替 ul 无序列表方案中的两个 div，从而通过标签的差异，再

角度,最终确定方案2。

## 案

完整它的样式代码。





```

        .arc-list p {
            height: 72px;
            line-height: 24px;
            color: #777;
        }
    </style>
</head>
<body>
    <div class="arc-list">
        <dl class="clearfix">
            <dt>
                
            </dt>
            <dd>
                <h2>经典的斐波那契数列与 arguments.callee</h2>
                <div>
                    <span>2016-08-15</span>
                    <span>文章类别: JavaScript</span>
                </div>
                <p>HTML5 学堂: 经典的斐波那契数列与 arguments.callee。提到斐波那契数列,
                很多人还不是太清楚,但是如果提到兔子的繁殖这个经典题目,相信学过计算机语言的人们会感觉
                很亲切。</p>
            </dd>
        </dl>
        <dl class="clearfix">
            <dt>
                
            </dt>
            <dd>
                <h2>WEBP 格式的图片</h2>
                <div>
                    <span>2016-07-30</span>
                    <span>文章类别: JavaScript</span>
                </div>
                <p>HTML5 学堂: 谷歌于 2010 年推出的新一代图片格式 —— WEBP,随着移动互联
                网的发展,WEBP 格式在 2015 年逐渐地开始被大公司部分采用。</p>
            </dd>
        </dl>
        <dl class="clearfix">
            <dt>
                
            </dt>
            <dd>
                <h2>使用 Git 多人协作,完成项目开发</h2>
                <div>
                    <span>2016-07-10</span>
                    <span>文章类别: 工具类文件</span>
                </div>
                <p>HTML5 学堂: 关于 Git 的知识,我们共分成了四个大步骤进行讲解,之前我们提
                到了 Git 的安装与配置、Git 在本地的使用方法,如何创建 Git 本地仓库与服务器端仓库的关系。</p>
            </dd>
        </dl>
    </div>

```



```
</dd>
</dl>
</div>
</body>
</html>
```

显示效果如图 5.18 所示。



图 5.18 实现样式控制优化后的标签选择实战效果图

#### 代码备注：

需要注意，出于显示效果的考虑，在此处给出了样式代码，但是此处重点讲解的是标签的选择，因此，请重点查看结构标签的变化。

应该说与其他方案相比，这套方案在各个方面都脱颖而出，均符合了基本规范与要求，但是，这样的代码能不能被真正的使用呢？又有哪些问题还没有考虑周全呢？让我们一起走入第 6 章，看看关于代码的可用性和扩展性。





## 第6章

# 模块布局(下)——可用性与扩展性



## 6.1 a 标签

### 6.1.1 超链接

超链接是互联网的核心技术,也是一个网站的灵魂,各个单独的网页之间,需要依靠 a 标签进行跳转,网页必须经过超链接连接之后,才能构成一个网站。

超链接,是指从一个网页地址指向另一个网页地址的链接关系,这个地址大部分情况下是一个网址,当然也可以是一个图片、邮箱地址、文件等。

### 6.1.2 超链接的属性

#### 1. href

href 属性是 a 元素最重要的属性,用于指示链接的目标,该属性对于 a 标签来说必不可少。

对于外部链接,通常使用完整的地址,包括链接所使用的协议(HTTP、HTTPS 等)以及网站地址。没有协议的链接是无效链接。

有效链接: `<a href="http://www.h5course.com/"> HTML5 学堂</a>`

无效链接: `<a href="www.h5course.com/"> HTML5 学堂</a>`

对于内部链接,通常使用相对路径,如: `<a href="../index.html"> HTML5 学堂</a>`。

#### 2. title

为了让用户能够单击某个区域跳转到其他相关页面,开发工程师会为其添加超链接,有时,网站开发方希望用户能够通过链接,看到这个链接的一些介绍信息,因此会为超链接设置 title 属性,当用户的鼠标移入超链接时,会显示 title 属性的具体属性值。

适当地为 a 标签添加 title 属性,除了能够提升用户体验之外,还能够提升网页的 SEO,对于搜索引擎爬虫来说,a 标签中的 title 属性也是要抓取的重要部分,前端工程师还可以通过在 title 属性当中书写多个关键词,提升“权重”。

代码实例:

```
<a href="http://www.h5course.com/" title="链接将跳转到 HTML5 学堂">HTML5 学堂 </a>
```



### 3. target

a 标签的 target 属性,用于控制链接页面的打开位置。在默认情况下,单击一个超链接之后,会自动刷新当前页面。

target 属性值如表 6.1 所示。

表 6.1 target 的属性值

值	描 述
_blank	浏览器总在一个新打开、未命名的窗口中载入目标文档
_self	默认状态,在当前页面/目标中打开被链接文档
_parent	使得文档载入父窗口或者包含来自超链接引用的框架的框架集
_top	清除所有被包含的框架并在整个窗口中打开被链接文档
framename	在指定的框架中打开被链接文档

#### 代码实例：

```
<a href = "http://www.h5course.com/" target = "_blank" title = "HTML5 学堂, h5course">HTML5 学堂 </a>
```

#### 相关说明：

所有属性值当中,使用最为频繁的是 \_self 和 \_blank,由于默认打开方式为 \_self,所以,通常在需要打开新页面时才会使用 target = "\_blank" 的属性设置。

对于 \_parent、\_top、framename,当前都已经不再使用,了解即可。

#### 框架：

在之前的开发方法当中,有时将一个网页拆分为多个部分,每个部分是一个网页,通过框架将多个网页组合到一起,于是就能够实现“单击左侧某个超链接,右侧某个框架中的页面发生变化”的修改。

在 HTML5.0 版本推出之前,在网站开发当中,还会时常使用到“框架”,即 < frameset >、< frame > 等标签。但在 HTML5.0 当中,原有的框架标签已经被废弃,而今的开发当中也不再使用了,在此就不详细讲解了。

## 6.1.3 锚链接

### 1. 什么是锚链接

锚链接是一种特殊的链接方式,它是在原有的链接基础之上,添加锚标记的后缀。通常用于实现调整到当前网页中的某一位置,或跳转到其他网页的某一位置。

### 2. 生活中的锚链接——书签

生活中,每个人都有书,假如希望能够快速访问到书籍的某一页,通常会在这个页的位置添加一个书签。

一个网页就如同一本书籍,我们希望能够快速查找到书籍的某个位置,就需要先创建一个书签,再将这个书签放置在书籍中对应的位置。在网页当中,通过 id 属性来定义一个锚链接(书签),这个锚链接可以添加在任何标签上。定义之后,找到需要“操作”锚链接的 a 标







签,在 a 标签当中设置 href 属性,属性值为:“#”加“锚链接名”,之后就可以通过单击 a 标签实现跳转了。

### 3. 锚链接的具体实例

(1) 需要跳转的地方添加代码: <a href="#命名">文字</a>。

(2) 需要在被转到位置添加代码: <标签名 id="命名"></标签名>。需要注意的是,标签中的 id 属性值与之前 a 标签中的命名必须相同。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 锚链接</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    .wrap {
      width: 400px;
    }
    .con {
      height: 800px;
      border: 1px solid black;
    }
    .nav a {
      float: left;
      width: 198px;
      height: 40px;
      border: 1px solid #39f;
      line-height: 40px;
    }
    .foot {
      border: 1px solid red;
    }
  </style>
</head>
<body>
  <div class="wrap">
    <div class="nav clearfix">
      <a href="#foots" title="内部锚链接">内部锚链接</a>
      <a href="锚链接 2.html#foots" title="外部锚链接">外部锚链接</a>
    </div>
    <div class="con">锚链接.html 文件</div>
    <div id="foots" class="foot">此处被设置了锚链接,单击顶部的导航能够直接使页面
    跳转到当前位置</div>
  </div>
</body>
</html>
```





4. 生活中的哪些网站运用了锚链接

锚链接在日常的企业网站开发当中并不常用,较多使用锚链接的网站,通常是一些开发手册、API 文档,如 HTML5 开发文档、Bootstrap 开发文档等。

6.1.4 超链接的基本样式

1. 伪类选择器

在网页当中,每种元素并不仅仅拥有一种状态。计算机为我们提供了鼠标、键盘等实现网页交互的工具,当用户与 HTML 网页文档进行交互时(如使用鼠标单击、光标移入某个元素等),元素就出现了普通状态之外的其他状态。或者可以将伪类选择器的功能理解为:在选中标签之后,为标签的某一种状态指定样式。

由于伪类选择器是对标签某种状态的控制,因此,并不能够单独使用,需要和其他选择器一起使用,普通选择器与伪类选择器组合在一起,能够选择到“某种标签的某种状态”。

例如: a:hover,就是光标移入 a 标签时状态的指定样式。

2. a 标签的伪类

a 标签的 4 种伪类选择器如表 6.2 所示。

表 6.2 a 标签的 4 种伪类选择器

伪类选择器	描 述
:link	超链接还未被访问的状态,也是超链接的默认状态,可以不设置
:visited	超链接已被访问过的状态
:hover	光标悬停在标签上的状态,不局限于 a 标签
:active	光标在标签按下时(标签被鼠标单击但还没有释放鼠标按键时)的状态,不局限于 a 标签

3. 伪类选择器的优先级

a 标签的伪类选择器优先级跟类的优先级是一样的,都是 0 0 1 0。

代码实例:

```
a:link {
    color: #f30;
}
a:visited {
    color: #000;
}
a:hover {
    color: #fff;
}
a:active {
    color: #f99;
}
```







#### 4. 伪类选择器的顺序

a 标签书写要求遵循如下顺序:link→:visited→:hover→:active。

伪类选择器的顺序不能颠倒的最主要原因在于,4种伪类选择器的优先级相同,由于网页文档的加载是自上而下的,因此后加载的样式会覆盖掉之前加载的样式,如果顺序错误,那么会导致在某些状态下的显示样式被其他状态的显示样式所覆盖。之所以是按照 link、visited、hover、active 的顺序进行书写,是由于:

(1) 当链接没有被访问的时候,显示 link 样式,也就是链接的默认样式。

(2) 当访问过了某一个链接,应当让用户能够看出和未访问链接的不同,因此,visited 伪类的样式应当覆盖 link 伪类的样式;可以得出书写顺序 link→visited。

(3) 当光标移动到链接上时,无论被访问过的标签还是没有被访问的标签,都应当出现光标移入即 hover 的样式效果,因此,hover 这个伪类的样式,必须能够覆盖掉 visited 和 link 的样式;可以得出 link→visited→hover。

(4) 当鼠标在标签上按下时,应当显示 active 样式。但是注意,当鼠标在标签上按下时,光标一直处于标签上,也就是说,hover 伪类的样式一直处于激活状态,那么要区分开“光标移入不按下鼠标”和“光标移入并按下鼠标”两种状态,就需要用 active 伪类样式覆盖掉 hover 的伪类样式。自此,可以得出这样的书写顺序:link→visited→hover→active。

#### 6.1.5 关于 a 标签的问题区

(1) a 标签当中,将 href 属性设置为 # 的目的何在?

a 标签设置 href="#",表示:为 a 标签设置一个链接,且链接为锚链接,但并没有指定具体要跳转到哪里。

这种方法可以快速实现单击之后返回顶部功能。

(2) a 标签当中,将 href 属性设置为 ### 或 javascript:void(0) 的目的何在?

在网站当中,无论是将 href 属性设置为 "javascript:void(0)",还是设置为 "###",这段代码的目的都是一个:创建一个超链接,但是,在单击链接时,既不能够跳转,也不会引起页面的刷新。

a 标签的 hover 是受到各个浏览器支持的,可以通过这些方法,实现一些按钮的样式。

这两种方法当中,设置为 javascript:void(0),相对会更好。

如果设置为 ###,其实表示的是为 a 标签设置了一个锚链接,锚链接的名称为 #。由于页面当中没有设置名为 "##" 的锚点,所以页面并不会发生跳转。但是,此时在 url 地址当中,会出现 ### 的内容。

网页当中,有些地方会涉及前后台数据的交互,而前端数据通常会通过 url 地址发送,当 url 地址的后面出现了 ### 时,就有可能对数据的传递产生影响,这也是这个方法的弊端。

代码实例:

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - a 标签 href 的不同属性值</title>
```





```
<link rel = "stylesheet" href = "../css/reset.css">
<style>
    .con {
        height: 1000px;
    }
</style>
</head>
<body>
    <a href = "###" title = "">在这个标签当中,设置 href 为###</a>
    <div class = "con">设置一个高度能够超出屏幕的元素,不然无法查看到效果</div>
    <p id = "##"> id 名为##,锚链接的值,在单击锚链接时,页面会滚动到这里</p>
    <a href = "#" title = "">此处的功能为单击之后返回页面顶端</a>
</body>
</html>
```

(3) a 的伪类样式是否一定要设置,如果不设置是什么样子?

a 标签有 4 种不同的状态,也可以对应 4 种不同的样式。但是,并不代表开发中 a 标签的这 4 种状态都需要进行设置。如果不设置 a 的样式或者只设置 a 的某些样式,未设置的样式会按照默认样式来显示。

谷歌浏览器中 a 标签 4 种状态的默认样式如表 6.3 所示。

表 6.3 a 标签 4 种状态的默认样式

常规-link 状态	访问后-visited 状态	光标移入-hover 状态	鼠标按下-active 状态
蓝色; 有下画线	紫色; 有下画线	和 hover 前一致	红色; 有下画线

注意: 在进行 a 标签默认样式测试时,不要引入 CSS 重置文件(在重置文件中设置了 a 标签样式); 测试时请注意浏览器缓存问题,测试之前需要清除缓存,否则有可能造成测试的结果有误; 另外,不同浏览器在默认样式中的状态可能有所不同。

(4) 当光标移入 a 标签时,希望 a 标签中的某些元素样式发生改变,应当如何书写?

在开发当中,有时会出现如下案例这样的需求。

希望实现的显示效果:

文章的标题

2016-09-20

结构代码:

```
<ul class = "arc_list">
    <li>
        <a href = "#" title = "">
            <strong>文章的标题</strong>
            <span>2016 - 09 - 20 </span>
        </a>
    </li>
</ul>
```

基本需求:

当光标移入 a 标签的时候,strong 元素中的内容,添加下画线。







此时,a 标签当中包含着两种标签,如果针对 a 进行 hover 伪类的设置,则 strong、span 元素并不会受到任何影响,因为操作的仅仅是 a 标签,而非 strong 和 span 标签。如下的代码设置不会对文本样式产生影响。

```
a:hover {  
    text-decoration: underline;  
}
```

此时,需要设置如下样式代码。

```
a:hover strong {  
    text-decoration: underline;  
}
```

此段代码中的选择器,其实是伪类选择器与后代选择器的“合体”,表示“当 a 标签处于 hover 状态时,a 标签中的 strong 标签的样式是什么”。

(5) a 标签能否嵌套块元素?

a 标签属于行元素,默认情况下是不建议嵌套块元素的,但是有时由于特殊需要(可触区方面的考虑)以及语义性的考虑,会为 a 标签设置 display: block;或 float,再使用 a 标签嵌套块元素。

a 标签嵌套的基本原则为:综合考虑 a 标签的可触区、样式的控制性、嵌套规则以及语义性,选择一种相对较为合理的嵌套方案。



## 6.2 光标样式

### 6.2.1 光标效果

在前端工程师进行页面开发时,针对光标的样式,会根据网页效果和用户体验的考虑,提出一些特殊要求。

例如,图中的 Tab 切换效果,为了让用户知晓“标题”是具有一定功能、可以单击的,会在 Tab 切换的标题区域针对光标样式进行处理,将光标样式处理成小手的状态,如图 6.1 所示。但是,这个标题区域的标签并非是 a 标签,单击之后也不能够跳转。

再比如在浏览京东、淘宝等电商型网站当中的具体商品时,当将光标移入到小图中时,右侧会显示出一张该图的大图,而在小图中的这个光标样式就会变成移动样式,如图 6.2 所示。

### 6.2.2 cursor 相关属性

光标样式主要通过 cursor 这个属性进行控制,



图 6.1 光标的小手样式





图 6.2 光标的移动样式

cursor 的具体属性值如表 6.4 所示。

表 6.4 光标样式 cursor 的属性值

值	描 述
cursor: crosshair;	十字准心
cursor: pointer;	手
cursor: wait;	等待/沙漏
cursor: help;	帮助
cursor: no-drop;	无法释放
cursor: text;	文字光标/编辑
cursor: move;	可移动对象
cursor: n-resize;	向上改变大小(North)
cursor: s-resize;	向下改变大小(South)
cursor: e-resize;	向右改变大小(East)
cursor: w-resize;	向左改变大小(West)
cursor: ne-resize;	向上右改变大小(North East)
cursor: nw-resize;	向上左改变大小(North West)
cursor: se-resize;	向下右改变大小(South East)
cursor: sw-resize;	向下左改变大小(South West)
cursor: auto;	自动
cursor: not-allowed;	禁止
cursor: progress;	处理中
cursor: default;	系统默认
cursor: url('../images/cursor.ani');	自定义光标样式

注意：

- (1) 能够为 a 标签改变其光标的默认效果；





(2) 能够为其他元素指定光标 hover 时的样式,而不仅局限于 a 标签;

(3) 在众多的光标样式属性当中,“手”(pointer)和“可移动对象”(move)在日常开发当中使用较多;

(4) 能够自定义光标样式,但是需要指定光标文件,光标文件要求格式后缀为 .cur 或 .ani。

### 6.2.3 关于 cursor: hand

如果是较早学习 HTML 和 CSS 的人会很熟悉 cursor 的这个属性值: hand。cursor: hand;和 cursor: pointer;的作用是一样的,都是将光标显示为小手状态,hand 这种属性值是 IE 单独发明并应用于 IE6 以前浏览器当中的,需要兼容低版本浏览器时才会使用,对于当前低于 IE6 版本的浏览器早已退市,这个属性值也就退出江湖了。

### 6.2.4 关于自定义光标样式的支持程度

所有主流浏览器虽然都支持 cursor 属性。但是对于自定义光标样式的支持不尽相同。

Firefox、Chrome 支持 jpg、gif、cur、png 等文件,不支持 .ani; IE 只支持 cur、ani; Opera 不支持自定义图标。IE6~IE8 对图标的大小有限制,最佳尺寸为 32×32(小尺寸图标会被拉伸,大尺寸图标会被压缩),其他浏览器按图标实际大小显示。



## 6.3 标签选择实战(3)——添加链接

### 6.3.1 为实例添加 a 标签

继续优化之前的案例(续 5.10),对于一个文章列表页,其主要作用是让用户查看,并能够跳转到具体的内容页面,之前方案中的代码如下,能够发现,并没有任何超链接能够让用户从列表页跳转到具体的文章内容页。

初始结构代码实例:

```
<div class="arc-list">
  <dl>
    <dt><img src="" alt="" title=""></dt>
    <dd>
      <h2>文章大标题</h2>
      <div><span>年月日</span><span>文字分类</span></div>
      <p>文章描述信息</p>
    </dd>
  </dl>
</div>
```

此时,虽然知道要添加链接,但是链接要添加给谁呢?或者说,a 标签的位置应该在哪里更合适呢?

在公司的实际开发当中,会有需求文档,告知开发工程师在哪个位置需要添加链接。如



果公司没有提供比较完善的文档,也不要急,根据自己平时浏览网页的习惯,来看看如何在这段代码当中添加超链接。

平时浏览网页,人们通常是通过单击“文章的预览图片”或“文章标题”实现页面跳转,那么在此处,可以将代码修改为如下结构。

添加 a 标签的结构代码实例:

```
<div class="arc-list">
  <dl>
    <dt><a href="" title=""><img src="" alt="" title=""></a></dt>
    <dd>
      <h2><a href="" title="">文章大标题</a></h2>
      <div><span>年月日</span><span>文字分类</span></div>
      <p>文章描述信息</p>
    </dd>
  </dl>
</div>
```

更改结构之后,需要注意如下两点。

- (1) 对于 a 标签的样式,能否进行控制,是否需要单独起类名?
- (2) a 标签存在多种样式,在书写时通常至少要提供默认样式和 hover 样式,而这些样式是由设计师在 PSD 图中或需求说明文档中给出的。

### 6.3.2 调整可触区

a 标签属于行元素,默认情况下,a 标签由内容撑开宽高。此时,就会出现 a 标签的可触区问题。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - a 标签的可触区</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    .tit {
      overflow: hidden;
      width: 400px;
      height: 40px;
      border: 1px solid black;
      line-height: 40px;
    }
  </style>
</head>
<body>
  <h2 class="tit">
```



```
<a href = "http://www.h5course.com" title = "">存在链接的文章标题</a>
</h2>
</body>
</html>
```

显示效果如图 6.3 所示。

图 6.3 a 标签的可触区

此时,当光标移入文字时,能够变成小手效果,单击之后能够实现链接跳转。但是在文字之外的其他空白区域,并不能够单击(即没有 a 标签效果)。

如果 a 标签能够充满 h2 标签的内容区,用户就更容易单击到链接,所以需要为 a 标签设置如下样式。

样式代码实例:

```
.tit a {
    display: block;
    height: 100%;
}
```

此时,a 标签充满了 h2 的内容区,光标移入到区域中,就会触发 a 的 hover 效果,单击之后就能够实现页面跳转。

### 6.3.3 当前方案的具体代码

样式代码方面,在原来基础之上,添加 a 标签的样式。

代码实例:

```
.arc-list dd a: hover {
    text-decoration: underline; //该语句表示光标移入时,文字出现下划线,在第 8 章当中会
                                //详细讲解
}
```

注:为了便于查看,在此给出第一个 dl 的结构,后面两个 dl 结构相同,在此省略。

结构代码实例:

```
<div class = "arc-list">
    <dl class = "clearfix">
        <dt>
            <a href = "#" title = ""><img src = "images/素材图片 (1).jpg" alt = "文章 1 - 配图" title = ""></a>
        </dt>
        <dd>
            <h2><a href = "#" title = "">经典的斐波那契数列与 arguments.callee</a></h2>
        </div>
```

```
<span>2016 - 08 - 15 </span>
<span>文章类别: JavaScript </span>
</div>
<p>HTML5 学堂: 经典的斐波那契数列与 arguments.callee。提到斐波那契数列,很多人还不是太清楚,但是如果提到兔子的繁殖这个经典题目,相信学过计算机语言的人们会感觉很亲切。</p>
</dd>
</dl>
</div>
```

备注: 如果在 a 标签的 href 属性当中不添加任何内容的话,并不会触发 a 的 hover 效果和功能,因此需要书写一个 # (锚链接),在后期前后台数据整合时,将 href 中的内容更换为实际链接地址即可。

6.3.4 方案结束了吗

截止到现在,我们考虑了很多方面的因素,选择了标签,完善了代码,这段代码能否算是结束了呢?

先做一些尝试: 更换 img 标签的图片路径(换成一张尺寸更大或更小的图片); 将文章的标题(h2 标签)更换为一个较长标题时; 将 p 标签中的文字内容多写一些时,页面会变成什么样子?

1. 当标题的文字较多时的页面效果

当标题文字较多时,标题换行,高度增大,导致右侧内容整体下移,从而布局错乱,如图 6.4 所示。

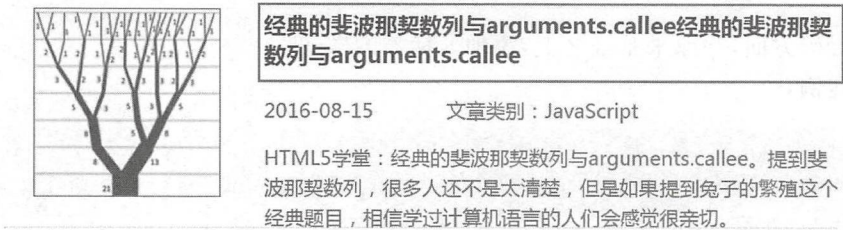


图 6.4 标题文字过多导致的布局错乱

此时需要为其设置固定高度以及超出隐藏。

2. 当图片被更换为较大尺寸图片时

当图像更换为一张大尺寸图片时,页面中的图像和文字重叠在了一起,如图 6.5 所示。究其原因在于: 虽然对 img 的容器进行了尺寸控制,但是并没有对 img 进行尺寸大小控制。

3. 当图片被更换为较小尺寸图片时

当图像更换为一张小尺寸图片时,虽然不会对页面的布局造成影响,但是图像的视觉效果方面也会存在问题,如图 6.6 所示。造成这种现象的原因和第二点相同,即没有对 img 进行尺寸大小控制。





图 6.5 图片更换为较大图片时导致的布局错乱

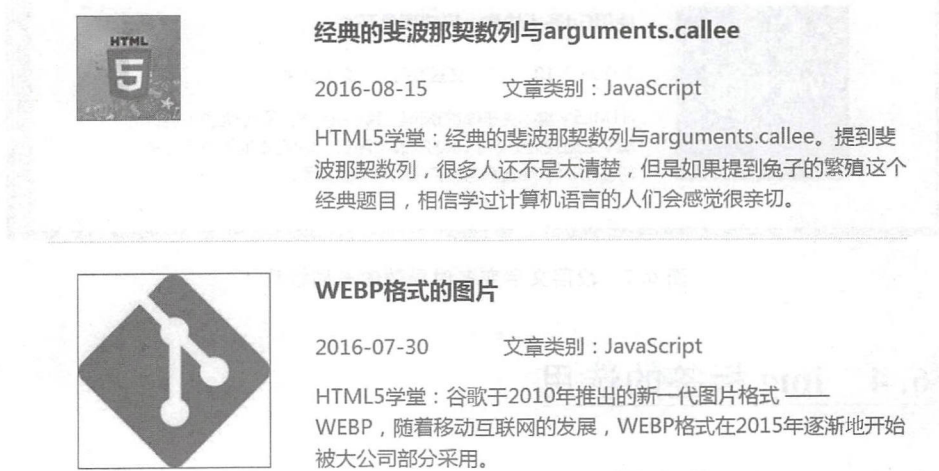


图 6.6 图片更换为较小图片时导致的视觉效果变差

4. 当段落内容较多时

当段落文字较多,也会造成文字溢出,从而导致布局和视觉效果受到影响,如图 6.7 所示。

5. 小结

看完上面的 4 种情况,相信你和我一样,整个世界都不好了!

有可能你会说,前端工程师开发时并不会把文字填写这么多,也不会用不正常尺寸的图。没错,作为前端开发工程师的我们,的确会按照正常的尺寸大小进行操作,但是数据是来源于后台的,后台的数据,无论数据量还是图像尺寸,都是有可能出现问题的,而作为前端工程师,则要尽可能避免这些问题。

当然,在真正的开发当中,想要解决这些问题并不难,不过需要先了解一些后台维护对前端的影响。

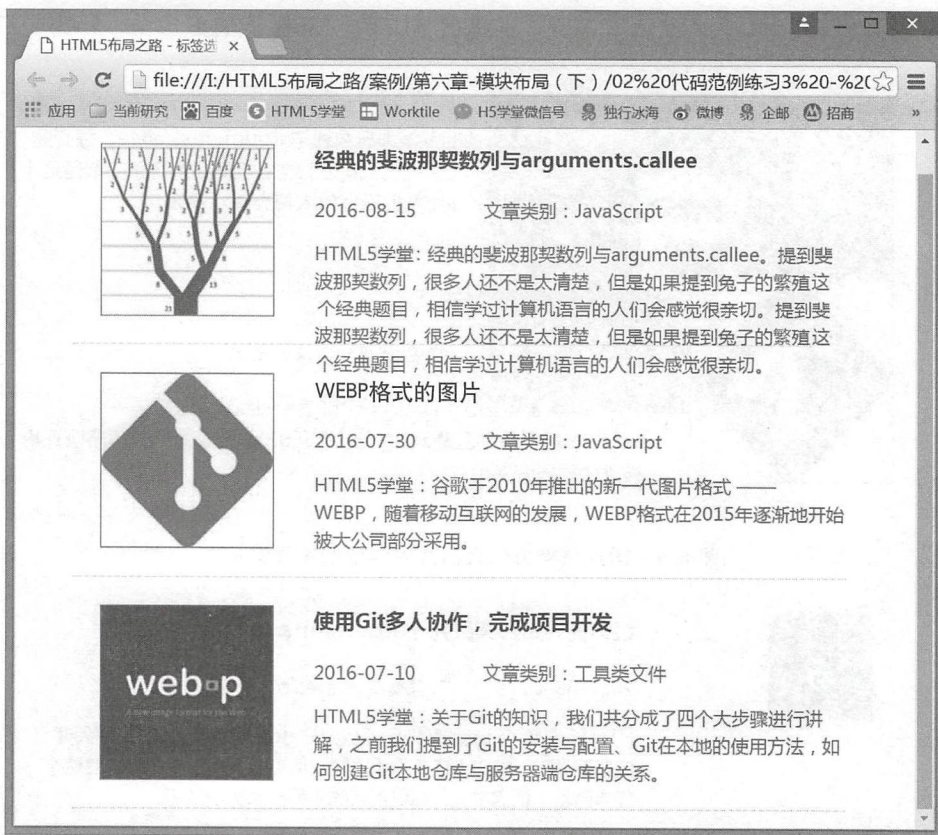


图 6.7 段落文字变多时导致的布局错乱



## 6.4 img 标签的选用

### 6.4.1 img 标签基本语法

#### 1. img 标签是什么

img 标签的作用：向网页中嵌入一幅图像。

从技术角度更严谨地说，<img>定义一个被引用图像的空间，之后通过该元素的 src 属性，将相应图像链接到这个页面显示出来。需要显示的图像是需要提前准备好的。

#### 2. img 标签的必备属性

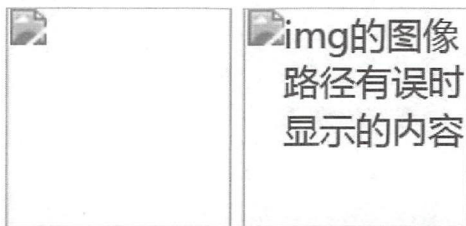
src 属性，用于设置图像的所在路径，src 属性和 href 属性在功能上其实是类似的，属性值都采用的是路径。alt 属性，用于规定图像的替代性文本。所谓替代性文本，指的是当网站速度较慢导致图像加载失败，或者图片链接出现错误时，替代原图片显示的具体文字。

如果没有书写 alt 属性，当图片没有显示出来时，会以一个红色的×号或断掉的小图片来显示，告知用户此处缺少一张图片（如图效果），但是对于用户来说，并不清楚这张图片是什么。



考虑更好的用户体验,为标签添加 alt 属性,当图片加载失败时,会将 alt 中的具体文本显示在 img 标签当中。

如图 6.8 所示效果图中,左侧图为没有书写 alt 属性的状态,右侧图为书写 alt 属性的状态。



为了便于查看,为img标签设置了  
100像素宽度,100像素高度

图 6.8 img 中的 alt 属性作用

## 6.4.2 数据图与背景图

### 1. 网页中的图片种类

在页面制作中,图片存在两大类,分别是数据图和背景图。

数据图,指的是从后台传递过来的图片,比如:列表页当中与每一个列表项相匹配的图片、文章页具体内容中的图片等。

背景图,指的是不需要从后台传递的图片,比如:各类小图标(icon)、LOGO 图片等。

在网站开发中,使用 img 标签存放数据图,使用标签的背景类样式进行背景图的处理。关于具体背景类的样式,将在 7.1 节当中进行详细的介绍。

### 2. 区分背景图和数据图

一个网站当中存在很多图片,并非所有的图片都是背景图。主要根据数据来源来区分背景图和数据图,通常来自于后台的数据使用数据图。

在很多新手刚开始网站开发时,很容易陷入如下两种误区当中:经常更换的图片是数据图,不经常更换的图片是背景图;大图片是数据图,小图片是背景图。

### 3. 图像类型选择练习

查看如图 6.9 所示这个网页界面,尝试对网页中的每种图片进行归类。

### 4. 图像类型选择练习答案

数据图与背景图的选择练习题答案如图 6.10 所示。

## 6.4.3 img 问题的规避

在默认情况下,img 的底部存在 3~5 像素的空隙(不同浏览器的具体像素值有所不同)。

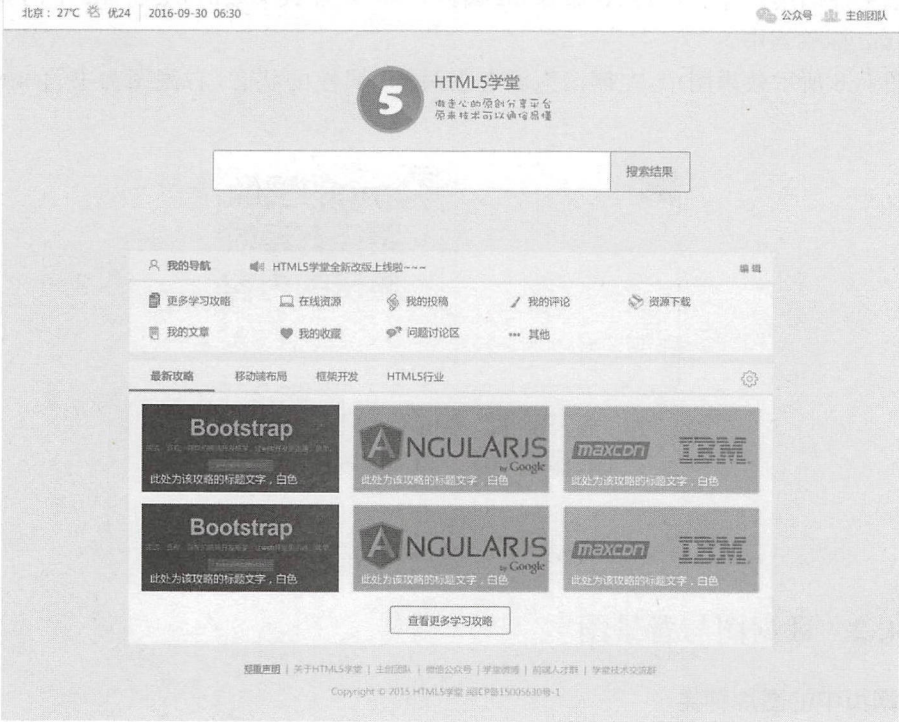


图 6.9 数据图与背景图的选择

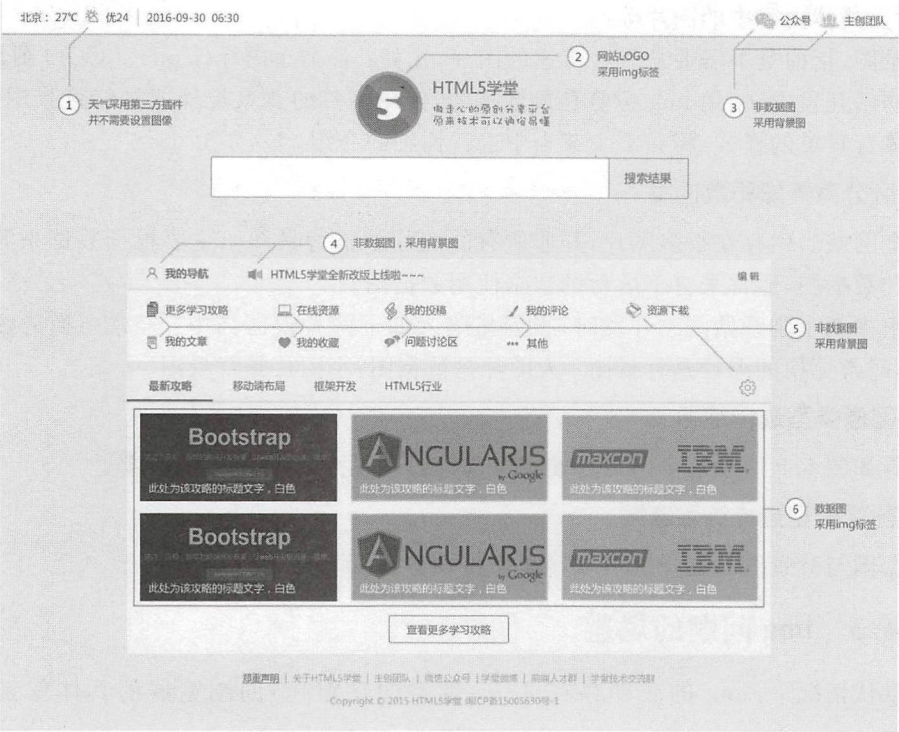


图 6.10 数据图与背景图的选择答案



代码范例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - img 底部的像素问题</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      border: 5px solid red;
    }
    .wrap img {
      width: 100px;
      height: 100px;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <img src = "../images/HTML5.jpg" alt = "HTML5" title = "">
  </div>
</body>
</html>
```

显示效果如图 6.11 所示。

当不为父级设置固定高度时,父级元素的高度由内容撑开,img 标签的高度为 100 像素,但是父级元素的高度为 104 像素(谷歌浏览器),说明在 img 标签下面存在着 4 像素的空隙。

最初刚出现 HTML 时,为了图文排版的可读性,默认为 img 底部设置了 3~5 像素的空隙。当前,需要清除掉 img 下的空隙,可以通过为 img 设置“float: left/right;”或“display: block;”来实现,从而防止 img 底部空隙对页面布局的影响。通常 display: block 会使用的更频繁一些,主要在于如果使用浮动来去除底部的空隙,会导致 img 脱离文档流,此时需要对 img 的父级元素清浮动,该方法相对比较复杂。

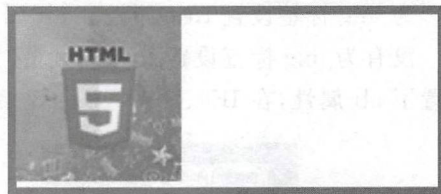


图 6.11 img 底部的 5 像素空隙

#### 6.4.4 img 中 alt 与 title 的区别

img 标签存在两种典型属性,alt 与 title。从含义、浏览器中的表现以及对于网站 SEO 优化程度三个方面,针对两者的特点进行比较。

示例代码：

```
<img src = "h5course.jpg" alt = "图像属性 alt 的值" title = "图像属性 title 的值">
```

当无法通过图片路径找到相应图片时,显示 alt 的属性值,当光标移入图片时显示 title 的属性值,如图 6.12 所示。

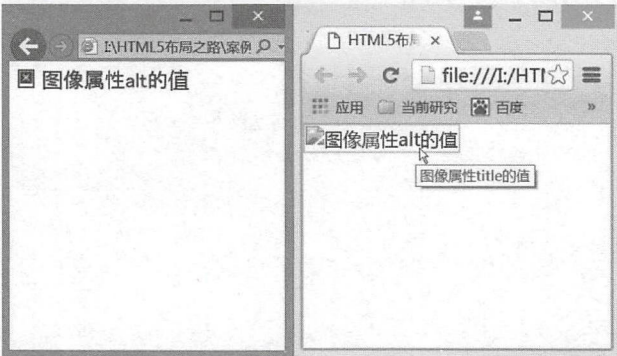


图 6.12 img 的 alt 与 title 属性

1. 具体含义方面

img 标签 alt 属性: 当图片不存在时或加载失败时的替代文字(进行显示)。

img 标签 title 属性: 对图片的描述与进一步说明。

2. 在浏览器中的表现方面

在 Firefox、Chrome 和 IE8+ 中,当光标经过图片时 title 属性的值会显示,而 alt 属性的值不会显示;在 IE6、IE7 中,如果 img 标签没有设置 title 属性,而只设置了 alt 属性时,在图片加载失败时显示 alt 的属性值;如果 img 同时设置了 title 属性和 alt 属性,在图片加载失败时显示 title 的属性值。

为 img 标签设置 title 属性和 alt 属性,在 IE6、IE7 中显示效果如图 6.13 所示。

没有为 img 标签设置 title 属性(注意,是不设置 title 属性,而不是 title 属性为空),但设置了 alt 属性,在 IE6、IE7 中显示效果如图 6.14 所示。

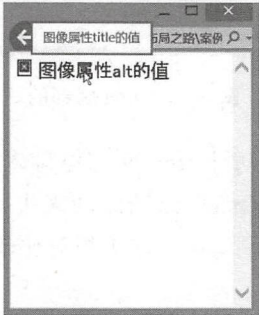


图 6.13 IE6、IE7 中 img 的 alt 与 title 显示 title



图 6.14 IE6、IE7 中 img 的 alt 与 title

3. 网站 SEO 优化方面

搜索引擎对图片含义的判断,主要靠 alt 属性,所以在图片 alt 属性中以简要文字说明,也是页面优化的一部分。条件允许的话,可以在 title 属性里进一步对图片说明。



### 6.4.5 href 与 src 的区别

#### 1. 含义角度

src(Source)属性仅嵌入当前资源到当前文档元素定义的位置。

href(Hypertext Reference)指定网络资源的位置,从而在“当前元素或者当前文档”和“href 属性定义的锚点或资源”之间定义一个“链接或者关系”。

#### 2. 解析角度

当浏览器找到标签之后,会读取src 的路径,之后浏览器进行图像的下载、提取和加载,在此之前,页面的加载和处理会被暂停。

对于 href,如:<link href="style.css" rel="stylesheet" />,此时浏览器解读出当前资源是一个样式表,页面解析并不会暂停(由于浏览器需要样式规则去画或者渲染页面,渲染过程可能会被暂停)。使用 link 标签 href 属性导入样式这种方式,能够加快网站的加载速度。



## 6.5 后台维护对前端的影响

### 6.5.1 图像加载对页面布局的影响

#### 1. 图像需要控制宽高

在后台维护方面,前端首要考虑的问题就是图像。对于一个网页来说,网页中的数据图是从后台申请过来的,主要由网站的运维人员,在后台管理系统当中进行图像上传等操作,有时运维成员会针对图像进行处理,将图像的大小修改为前端的规定尺寸,但是也有可能没有修改图片大小,直接将图像上传到后台管理系统当中。

此时,图像的初始大小和前端展示的图像大小可能会不相符。无论图像的初始大小“小于”还是“大于”前端规定的图像大小,都会影响img 标签的盒模型属性,从而进一步对网页的布局造成影响。

对于前端的我们来说,要尽可能地规避这个问题。规避的方式比较简单,为img 设置固定的宽度和高度即可,此处可以为img 标签设置具体的像素值,也可以与其父级元素挂钩,如果img 标签的父级元素有固定的宽度和高度,且img 标签需要占满父级元素的全部,可以对img 标签使用百分比这种相对度量单位,如图6.15所示。

#### 2. 数据图像变形的责任归属

有些人可能对于图像产生的“拉伸变形”耿耿于怀,各位不必担心,“数据图像变形”属于后台维护人员的责任范畴。网站的开发方,在工程项目完工时,需要将功能说明文档提交给后台维护人员,在文档中,会标明相应数据图的尺寸大小以及比例要求。

如果后期维护人员上传了错误比例的图像,就是维护人员自己的问题,而与前端开发人员没有关系。但是,如果是前端人员没有设置固定大小,而导致维护人员上传的图像超出(维护人员有可能上传的是比例正确但稍微大一点儿的图,也有可能切图时有1~5像素

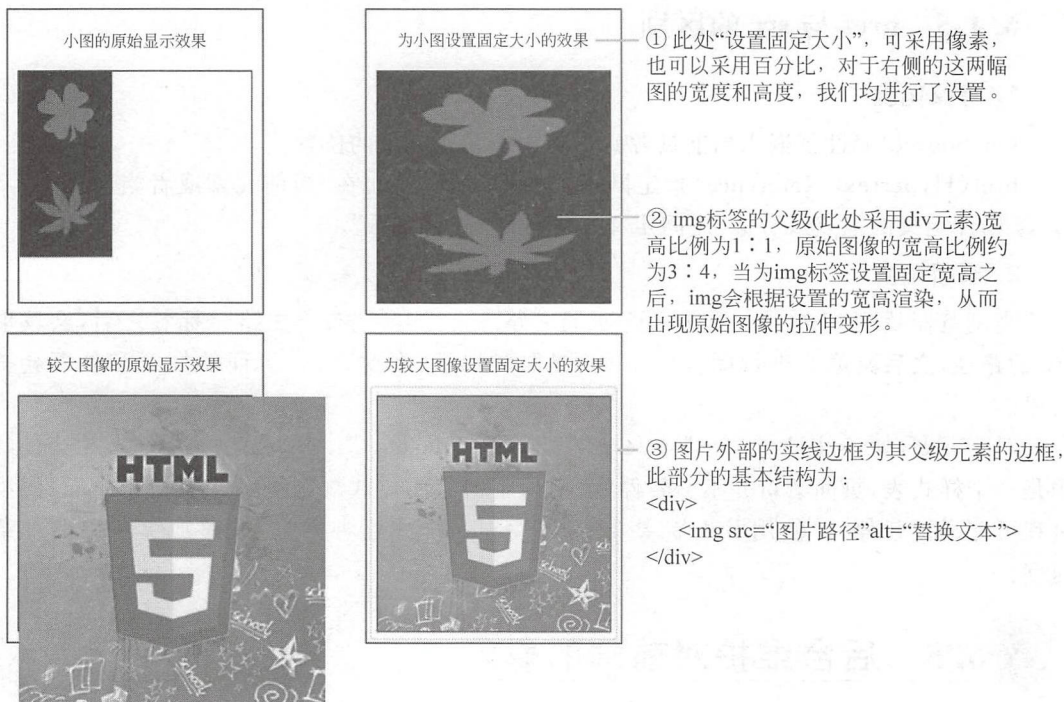


图 6.15 为图像设置大小之后的效果

的误差),那么责任在于前端开发人员。

### 3. 只设置 img 标签一个方向的尺寸值是什么效果

该方向会按照指定的尺寸值渲染,而另一个方向会根据指定方向的值等比例放大。

比如:原图像尺寸为  $150\text{px} \times 200\text{px}$ 。此时设置宽度为  $300\text{px}$ ,则高度自动变化为  $400\text{px}(300/150 \times 200)$ 。

## 6.5.2 文字超出造成的页面混乱

### 1. 超出的文本部分需要控制

除了图像大小的控制之外,前端要考虑的第二个问题,是文本内容。网页当中显示的文字,大部分来源于后台,后台传递过来的文字量有可能小于或大于前端元素能够容纳的文字量,此时,前端也需要进行处理。防止文本量不同时对布局的影响或对其他标签显示效果的影响。

文字量较小,元素无具体高度;文字量较小,元素设置具体高度;文字量较大,元素无具体高度;文字量较大,元素设置具体高度。在这4种情况下,分别是怎样的显示效果呢?一起来看如下代码。

代码实例:

```
<!doctype html>
<html>
<head>
```



```

<meta charset = "UTF - 8">
<title>HTML5 布局之路 - 文本超出造成的问题</title>
<link rel = "stylesheet" href = "../css/reset.css">
<style>
    .wrap {
        width: 880px;
        margin: 0 auto;
        font-size: 14px;
    }
    .wrap div {
        float: left;
        width: 200px;
        margin: 0 10px;
    }
    .wrap h1 {
        line-height: 36px;
        text-align: center;
    }
    .wrap p {
        border: 1px solid #f00;
        line-height: 20px;
    }
    .state1 p, .state3 p {
        /* 不设置具体高度 */
    }
    .state2 p, .state4 p {
        height: 100px;
        /* 设置具体高度 */
    }
</style>
</head>
<body>
    <div class = "wrap">
        <div class = "state1">
            <h1>文字量较小 & 元素无具体高度</h1>
            <p>前端要考虑的第二个问题,是文本内容。</p>
        </div>
        <div class = "state2">
            <h1>文字量较小 & 元素设置具体高度</h1>
            <p>前端要考虑的第二个问题,是文本内容。</p>
        </div>
        <div class = "state3">
            <h1>文字量较大 & 元素无具体高度</h1>
            <p>前端要考虑的第二个问题,是文本内容。网页当中显示的文字,大部分来源于后台,后台传递过来的文字量有可能小于或大于前端元素能够容纳的文字量,此时,前端也需要进行处理。防止文本量不同时对布局的影响或其他标签显示效果的影响。</p>
        </div>
    </div>

```

```

<div class="state4">
    <h1>文字量较大 & 元素设置具体高度</h1>
    <p>前端要考虑的第二个问题,是文本内容。网页当中显示的文字,大部分来源于后台,后台传递过来的文字量有可能小于或大于前端元素能够容纳的文字量,此时,前端也需要进行处理。防止文本量不同时对布局的影响或对其他标签显示效果的影响。</p>
</div>
</div>
</body>
</html>

```

显示效果如图 6.16 所示。

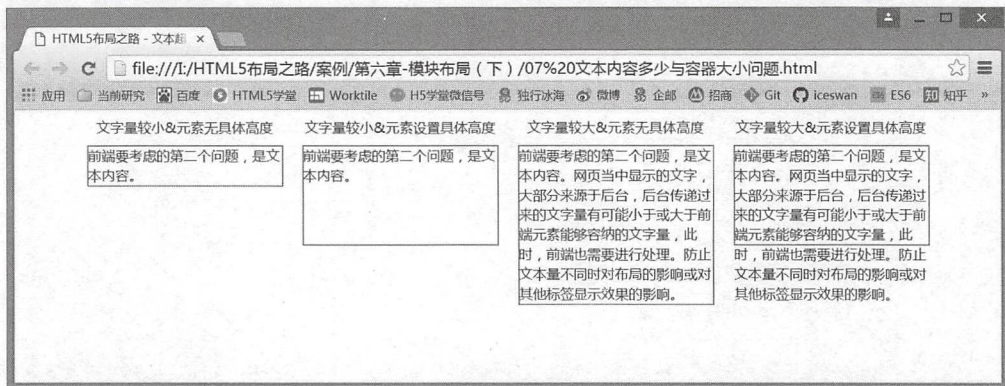


图 6.16 文本量与容器宽高限制的显示效果

#### 代码解析:

如果不设置具体高度,元素的高度是由内容撑开的(第一、三种显示效果)。

如果设置具体高度,当内容高度小于设置的高度值时,剩余高度区域留白(第二种显示效果);当内容高度大于设置的高度值时,内容会溢出,但是元素的盒模型保持不变(第四种显示效果)。

### 2. 需要明确的核心问题

可以得出一个基本结论:解决文字超出的问题并不难,只需要将其隐藏即可。但是,必须要明确几个核心点:对于文字的容器(即标签),哪些需要设置固定宽高,哪些标签可能需要考虑超出隐藏;如果需要隐藏,又应该如何书写命令,实现隐藏的效果和功能?



## 6.6 网页中哪里需控制高度或超出隐藏

### 6.6.1 不同页面的不同需求

对于是否设置固定高度,根据具体情况的不同,答案也有所不同,一个容器(即标签)是否需要设置固定宽高,是根据这个容器(标签)的具体需求而定。下面一起来分析一下网站中的几种典型页面。



1. 首页&二级页

如图 6.17 所示,网站的首页和二级页,大都是由各个模块组成的,对于页面中的每个模块,布局通常是规定死的,它们的位置、大小通常都有很明确的要求。对于这种页面中的固定模块,都为其设置固定的大小,虽然父级元素的高度也能够通过内容而撑开,但是设置固定高度相对会更保险一些,毕竟一旦内容高度在布局中出现问题,会对其父级元素造成影响,从而影响整个网页的布局。

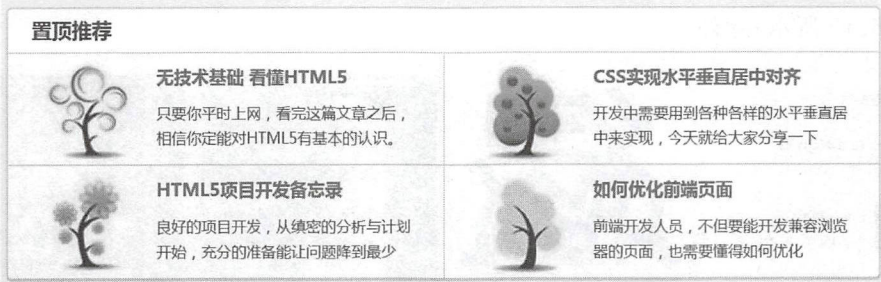


图 6.17 首页与二级页的一部分

2. 列表页

如图 6.18 所示,右侧为相应的文章列表。对于列表页来说,需要明确两个部分的高度设置。第一,包含列表的父级元素的高度。第二,每个列表项的高度。

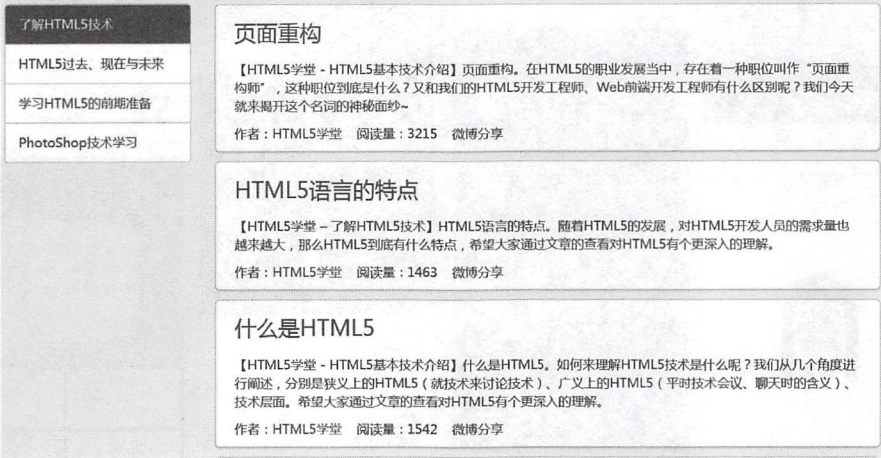


图 6.18 列表页的一部分

1) 列表项父级元素高度

实例当中,右侧有三条数据(文章),那么包含这三条数据的父级元素,高度是否能够定死呢?

在这里,数据条数是来源于后台控制的,后台可能给出的是显示 8 条数据的需求,但是在数据库当中只有三条数据,于是这里就只能展示三条数据。当数据库中数据大于三条时,会展示的更多(最多为后台定义的 8 条)。当然,后台在网站上线之后,也有可能进行更改,例如:感觉每页 8 篇文章稍微少了一些,调整为每页展示 10 篇文章。

如果此时为列表项的父级元素设置了固定高度,则需要进行前端页面的修改,如果父级元素的高度默认为内容撑开,前端页面就不需要进行任何的样式调整,相对来说代码的扩展性较好。

## 2) 每个列表项的高度

出于显示效果的一致性,通常情况下,每个列表项的高度是一致的,从这张图中也能够查看到,有的文章介绍性的文字有三行,有的文章介绍性文字只有两行,但是这些列表项高度都是一致的。因此,大部分情况下,会为每个列表项设置固定高度。但是,瀑布流式的布局(如图 6.19 所示)除外。



图 6.19 瀑布流布局

## 3. 内容页

图 6.20 是一个较为常见的内容页。文章内容页存在的主要目的在于“完整地”展示信息,无论是文章的标题、关键词,还是文章的具体内容,都应当让其尽可能完整地展示出来,因此,在这里并不能够限制内容页的具体内容区域,即不针对内容区设置高度。





图 6.20 内容页的一部分

## 6.6.2 关于高度设定的基本结论

网站开发的大部分情况下:

- (1) 首页和二级页的模块部分, 设置固定宽高。
- (2) 列表页中, 包含列表项的父级元素, 不能够设置固定高度。
- (3) 列表页中, 每个列表项高度有限, 需要设置固定高度。
- (4) 内容页中, 标题以及内容区, 不能够设置固定高度, 应当由内容撑开高度。

## 6.6.3 何时考虑超出隐藏

通常是在考虑后台对前端影响的时候, 要针对超出部分进行处理。很多时候数据是从后台传到前端页面当中的, 那么有时有些地方内容会比较多, 有些地方内容会比较少。下面针对这种情况做个简单的整理。

- (1) 对于 `img` 元素的父级标签, 建议设置超出隐藏;
- (2) 对于列表页的标题和内容描述部分, 通常需要针对超出进行设置; 多行的内容描述部分需要设置超出隐藏; 单行的列表页标题, 可以设置超出隐藏或超出显示为省略号;
- (3) 对于内容页的标题和内容, 千万不要随意设置固定高度, 也不需要设置超出隐藏。

## 6.6.4 关于“高度控制与超出隐藏”的问题区

- (1) 想要设置超出隐藏的前提是什么?

如果需要为一个元素设置超出隐藏, 当前元素必须有固定范围。

每个元素有默认的宽度, 但是, 当该元素没有设置固定高度时, 默认为内容撑开高度。随着内容的增加, 高度值也会发生动态的改变, 此时如果不进行特殊样式设置, 是不会出现

内容超出元素的现象的,也就无法触发超出隐藏。关于特殊样式设置,在 6.8 节当中会详细讲解。

(2) img 添加固定宽高之后,是否一定要给父级标签添加超出隐藏?

原则上来说,不需要。

为 img 标签设置固定宽高、为 img 标签的父级元素设置超出隐藏,这两种操作的目的是统一的,那就是防止后台数据传递时出现的问题。

两者都设置之后,相当于是双保险。在开发当中,img 的固定宽高应当设置,而 img 元素的父级元素需要根据具体情况来决定是否添加超出隐藏。如果在父元素当中有除 img 标签之外的其他元素,且任意子元素超出了父元素显示区域,则不能够为父级设置超出隐藏。



## 6.7 内容的超出处理——overflow

### 6.7.1 基本语法与功能

属性功能:

设置文本超出容器(标签)时的显示方式。

基本语法:

```
overflow: hidden;
```

代码解析:

设置当文本超出元素的显示范围时隐藏。

属性值如表 6.5 所示。

表 6.5 overflow 的属性值

值	描 述
visible	默认值。内容不会被修剪,会呈现在元素框之外
hidden	内容会被修剪,并且其余内容是不可见的
scroll	内容会被修剪,但是浏览器会显示滚动条以便查看其余的内容
auto	如果内容被修剪,则浏览器会显示滚动条以便查看其余的内容
inherit	规定应该从父元素继承 overflow 属性的值

x 和 y 两个方向的控制:

overflow 属性的设置是针对水平、垂直两个方向的,如果只想针对其中一个方向设置隐藏方式,可以使用 overflow-x 或 overflow-y,这两种属性的取值和 overflow 的取值相同。

### 6.7.2 实现文本超出隐藏

使用 CSS 实现元素的文本超出隐藏,通常存在两种方式,一种是超出直接隐藏内容,另一种是超出显示为省略号。

超出隐藏的代码比较简单,只需要为一个有固定宽高的元素设置为 overflow:hidden。



代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 文本超出隐藏</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      overflow: hidden;
      width: 200px;
      height: 40px;
      border: 2px solid black;
      line - height: 40px;
    }
  </style>
</head>
<body>
  <div class = "wrap">HTML5 布局之路 - 文本超出隐藏</div>
</body>
</html>
```

显示效果如图 6.21 所示。

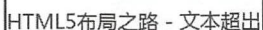


图 6.21 单行文本超出隐藏

### 6.7.3 实现文本超出显示为省略号

#### 1. 单行文本超出显示为省略号

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 文本超出显示为省略号</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .text - overflow {
      width: 400px;
      height: 40px;
      border: 2px solid black;
      line - height: 40px;
      overflow: hidden;
      text - overflow: ellipsis;
    }
  </style>
</head>
<body>
  <div class = "text - overflow">HTML5 布局之路 - 文本超出显示为省略号</div>
</body>
</html>
```

```

        word-break: keep-all;
        white-space: nowrap;
    }
</style>
</head>
<body>
    <div class="text-overflow">HTML5 布局之路。HTML5 学堂：本文当中我们主要为大家讲解
    如何实现文本超出显示为省略号；同时讲解一下，在网页开发与制作的时候，我们什么时候应该考虑
    内容撑开宽高，又应该在何时考虑文本超出的问题。</div>
</body>
</html>

```

#### 代码解析：

样式最后 4 行的代码是超出隐藏显示为省略号的“核心代码”。

overflow: hidden;表示的是“内容超出宽度时隐藏超出部分”。

text-overflow: ellipsis;表示当对象内文本溢出时显示省略标记(…);它必须与 overflow:hidden;一起使用。

word-break: keep-all;和 white-space: nowrap;的作用一致，都是让对象内的文本不换行。

## 2. 多行文本超出显示为省略号

多行文本超出显示为省略号的需求，可以由后台直接进行控制，也可以由前端开发工程师来实现。

对于后台来说，需要截取相应的数据量大小，之后在一段文字的最后，添加“…”的符号，再作为实际数据呈现在前端书写好的结构标签当中。这种方法，前端工程师并不需要做任何其他工作，当然，前端工程师也可以为相应的结构标签添加 overflow: hidden,从而达到双保险。

如果单纯由前端工程师来实现这个需求，那么就稍微有些复杂了，如果仅使用 HTML 和 CSS 很难实现，通常会借助 JavaScript 来辅助实现。

#### 代码实例：

```

<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 多行文本超出显示为省略号</title>
    <link rel="stylesheet" href="../css/reset.css">
    <style>
        .text-overflow {
            width: 400px;
            border: 2px solid black;
            line-height: 20px;
        }
    </style>
</head>

```



```

<body>
  <div class="text-overflow" id="con">HTML5 布局之路。HTML5 学堂：本文当中我们主要为
  大家讲解如何实现文本超出显示为省略号；同时讲解一下，在网页开发与制作的时候，我们什么时候
  应该考虑内容撑开宽高，又应该在何时考虑文本超出的问题。</div>
  <script>
    var con = document.getElementById('con');
    var textCon = con.innerHTML;
    con.innerHTML = textCon.substr(0, 49) + '...';
  </script>
</body>
</html>

```

#### 代码解析：

前端实现的基本原理与后台类似，也是进行字符串截取并连接一个“...”。

该段 JavaScript 代码的含义是：通过 innerHTML 获取元素的内容，之后使用字符串方法进行截取，截取前 49 个字符，再在这 49 个字符之后连接一个“...”，最后，将这个截取后的字符串赋值给原来的元素内容。

195

### 6.7.4 关于 overflow 的问题区

(1) overflow: scroll; 与 overflow: auto; 的区别？

当文本超出时，overflow: scroll; 与 overflow: auto; 都会出现滚动条。

当文本没有超出元素区域时，overflow: auto; 并不会出现滚动条，而 overflow: scroll; 依旧会显示滚动条。

(2) 超出隐藏还是省略号，由谁来确定？

文本超出时需要隐藏还是显示为省略号，主要是根据网站需求而定的，如果在开发过程中对需求不够清楚的话，可以与产品经理沟通，明确需求。



## 6.8 代码扩展性——关于 margin 负值

### 6.8.1 前后台数据整合方式

前端书写的页面仅仅是结构样式和模拟的数据，并不能够直接上线让用户查看。在上线之前，需要进行前后台数据整合，让前端页面展示实际的数据（来自于数据库）。在整合过程中，前端书写的相似内容会被替换掉。

#### 1. 前端书写的代码

代码实例：

```

<body>
  <div class="arc-list">
    <dl class="clearfix">
      <dt>

```

```

        <a href = "#" title = ""><img src = "images/素材图片 (1). jpg" alt = "文
章 1 - 配图" title = ""></a>
      </dt>
      <dd>
        <h2><a href = "#" title = "">经典的斐波那契数列与 arguments.callee </a>
      </h2>
      <div>
        <span>2016 - 08 - 15 </span>
        <span>文章类别: JavaScript </span>
      </div>
      <p>HTML5 学堂: 经典的斐波那契数列与 arguments.callee。提到斐波那契数列,
      很多人还不是太清楚,但是如果提到兔子的繁殖这个经典题目,相信学过计算机语言的人们会感觉
      很亲切。</p>
    </dd>
  </dl>
  <!-- 此处还有两个 dl,出于内容量的考虑,在这里省略掉没有书写 -->
</div>
</body>

```

如上为之前书写的前端结构代码。

## 2. 后台整合之后的代码

后台存在 PHP、Java、ASP 等各类语言,此处以 PHP 为例来书写后台整合后的代码。其他后台语言在整合时,原理与如下代码类似,只是基本语法有所不同。

代码实例:

```

<body>
  <?php
    $conn = mysql_connect("localhost", "root", "");
    mysql_query("SET NAMES 'utf8'");
    mysql_select_db("html5road");
  ?>
  <div class = "arc - list">
    <?php
      $sql = mysql_query("SELECT * FROM listdata LIMIT 0, 5");
      while ( $arr = mysql_fetch_assoc( $sql) ) {
    ?>
    <dl class = "clearfix">
      <dt>
        <a href = "arc.php?aid = <?php echo $arr['id'] ?>" title = "<?php echo
$ arr['title']; ?>">
          <img src = "<?php echo $arr['pic']; ?>" alt = "<?php echo $arr['title']; ?>">
        </a>
      </dt>
      <dd>
        <h2><a href = "arc.php?aid = <?php echo $arr['id'] ?>" title = "<?php echo
$ arr['title']; ?>"><?php echo $arr['title']; ?></a></h2>
        <div>
          <span><?php echo $arr['data']; ?></span>
          <span>文章类别: <?php echo $arr['kinds']; ?></span>
        </div>
      </dd>
    </dl>
  </div>

```



```
<p><?php echo $ arr['inf']; ?></p>
</dd>
</dl>
<?php } ?>
</div>
</body>
```

备注：

- (1) 此文件的后缀名为.php。
- (2) 代码需要在后台环境(即服务器环境)下运行,可以使用 Wamp 软件。
- (3) 此处只给出了 body 标签中的代码,head 标签当中依旧是引入的 CSS 以及样式代码,没有发生变化,因此不再重复书写。
- (4) PHP 这段代码是依托于服务器与后台的,在计算机上如果单纯地书写这段代码,并没有办法看到实际的效果。

3. 后台的数据表的内容

后台数据表的具体内容如图 6.22 所示。

id	title	pic	inf	data	kinds
1	经典的斐波那契数列与 arguments.callee	images/pic1.jpg	HTML5 学堂：经典的斐波那契数列与 arguments.callee。提到斐波那契数列，很多人还不是...	2016-08-15 00:00:00	JavaScript
2	WEBP 格式的图片	images/pic2.jpg	HTML5 学堂：谷歌于 2010 年推出的新一代图片格式——WEBP，随着移动互联网的发展，WEBP...	2016-07-30 00:00:00	JavaScript
3	使用 Git 多人协作，完成项目开发	images/pic3.jpg	HTML5 学堂：关于 Git 的知识，我们共分成了四个大步骤进行讲解，之前我们提到了 Git 的安装与配置、...	2016-07-10 00:00:00	工具类文件

图 6.22 后台数据表的具体内容

6.8.2 比数据条数少一个的虚线如何实现

在如图 6.23 所示的这张图当中,能够看到的是,每两个列表项之间会存在一条虚线(共三个列表项,但是只有两条虚线)。而具体实现的样式是,三个列表项,每个列表项的后面都



图 6.23 “比数据条数少一个的虚线”效果

有一条虚线。当前的显示效果与需求并不相符合。应当如何处理呢?

此时,相信你能够想到为第一个或最后一个标签设置一个类名,去掉那条虚线的样式。通过这种方式解决这种特殊需求。

在此将面临两个问题:

(1) 页面中列表项的数量不确定,是随着后台数据的数据而变化的,特殊类名要放在什么位置,虚线应当归属于哪个元素,值得慎重考虑;

(2) 如果多起一个类名,会让后台的“循环”变得不可执行。

### 6.8.3 特殊情况类名设置详析

(1) 将虚线设置为每个 dl 元素的下边框,并且为最后一个 dl 元素设置特殊样式的类名。

代码实例:

```
<div class="arc-list">
  <dl class="clearfix">
    <!-- 在此省略内容部分 -->
  </dl>
  <dl class="clearfix">
    <!-- 在此省略内容部分 -->
  </dl>
  <dl class="clearfix special">
    <!-- 在此省略内容部分 -->
  </dl>
</div>
```

问题解析:

前端代码看上去虽然可行,但是当后台进行代码整合时,代码变得不可用。由于后台的数据条数不确定,也就不确定到底有多少个列表项会显示在页面当中,此时,后台没有办法进行最后一个 dl 的标签遍历。

当然有可能你也会想到“将虚线设置为每个 dl 元素的上边框,并且为首个 dl 元素设置特殊样式的类名”。这种方法在后台整合代码时,需要留下前端中两种不同的 dl,分别进行遍历,从这个角度来说,前端书写的代码扩展性不够,增大了后台人员的工作量。

(2) 运用特殊的选择器。

代码实例:

```
.arc-list dl {
  border-top: 1px dotted #ccc;
}
.arc-list dl:first-child {
  border: none;
}
```

在选择器当中,存在:first-child 等伪类选择器(注意不是伪元素),该选择器选择到当前元素中的第一个子元素(.arc-list dl:first-child 表示的是选择到类名为 arc-list 元素中的第







一个 dl 子元素)。这种方法并没有给标签起不同的类名,也不会对后台数据整合造成任何影响。可以通过这种方式进行样式控制,但是,最大的问题在于伪类选择器在 IE6-浏览器中不兼容,即 IE6 中: first-child 选择器不能够使用。不过,当前的开发当中,很少有项目还需要兼容 IE6 了。

## 6.8.4 扩展性曾经的救世主——margin 负值

### 1. margin-top 负值实现需求

上面讨论了三种方法,后两种方法虽然都能够实现需求,但是都存在一定的弊端和问题,要么是扩展性太差,要么是部分特殊浏览器存在兼容问题且产生了额外的样式代码。

此时我们引入一个新的方法,该方法利用 overflow 配合盒模型的 margin 属性来实现。先利用这两种属性处理一下前端代码,查看一下效果,之后再做具体解释。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - margin-top 的负值来实现需求</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    .arc-list {
      overflow: hidden;
      width: 676px;
    }
    .arc-list dl {
      height: 200px;
      margin-top: -1px;
      border-top: 1px dotted #ccc;
    }
    /* 在此省略其他 CSS 样式 */
  </style>
</head>
<body>
  <div class="arc-list">
    <dl class="clearfix">
      <!-- 在此省略内容部分 -->
    </dl>
    <dl class="clearfix">
      <!-- 在此省略内容部分 -->
    </dl>
  </div>
</body>
</html>
```

代码解析:

在显示效果当中,最上面的边框被隐藏。





在这段代码当中,为父级元素设置超出隐藏(overflow: hidden),为父级元素中的每个dl列表元素,设置"margin-top: -1px"的命令,并通过设置dl元素顶部边框的方式实现虚线效果。

在不设置margin-top之前,几个dl元素正常地进行排列,最顶部是第一个dl元素的顶部虚线边框。此时,通过margin负值的设置,让每个dl元素向上运动1像素的距离,那么,第一个dl元素的虚线边框就运动出了父级元素的内容区域。在默认情况下,超出内容区的部分会正常显示。但是,我们并不希望它被浏览者看到,因此为父级元素设置超出隐藏,就让内容区之外的部分隐藏掉了。

2. 父级内边距可能的影响

1) 为何父级内边距会有影响

如果父级元素存在内边距,子级元素会从父级内容区开始显示,此时子级设置margin负值,内容会首先进入父级元素的内边距,依旧会被显示出来,并不会被隐藏。将上节代码按照表6.6进行调整。

表 6.6 父级 padding 值调整

调整前	调整后
<pre>.arc-list {   overflow: hidden;   width: 676px; }</pre>	<pre>.arc-list {   overflow: hidden;   width: 676px;   padding: 14px 50px; }</pre>

这种情况下,第一个dl顶部的边框就不会被隐藏,会显示在父级元素的padding区域当中。

2) 解决办法1——多嵌套一层标签

在当前的父级元素与子级元素之间嵌套一层元素,这层元素本身是从父级元素的内容区开始显示的,并不会存在内边距,因此,为新增元素设置超出隐藏即可,如表6.7所示。

表 6.7 嵌套一层标签之后的代码对比

样式代码调整前	样式代码调整后
<pre>.arc-list {   overflow: hidden;   width: 676px;   padding: 14px 50px; }  .arc-list dl {   height: 200px;   margin-top: -1px;   border-top: 1px dotted #ccc; }</pre>	<pre>.arc-list {   overflow: hidden;   width: 676px;   padding: 14px 50px; }  .arc-con {   overflow: hidden; }  .arc-list dl {   height: 200px;   margin-top: -1px;   border-top: 1px dotted #ccc; }</pre>







续表

结构代码调整前	结构代码调整后
<pre> &lt;div class = "arc - list"&gt;   &lt;dl class = "clearfix"&gt;     &lt;!-- 在此省略内容部分 --&gt;   &lt;/dl&gt;   &lt;dl class = "clearfix"&gt;     &lt;!-- 在此省略内容部分 --&gt;   &lt;/dl&gt; &lt;/div&gt; </pre>	<pre> &lt;div class = "arc - list"&gt;   &lt;div class = "arc - con"&gt;     &lt;dl class = "clearfix"&gt;       &lt;!-- 在此省略内容部分 --&gt;     &lt;/dl&gt;     &lt;dl class = "clearfix"&gt;       &lt;!-- 在此省略内容部分 --&gt;     &lt;/dl&gt;   &lt;/div&gt; &lt;/div&gt; </pre>

### 3) 解决办法 2——使用外边距替代内边距

可以使用外边距来替代内边距,根据不同情况,可以用子级的外边距替代父级的内边距,也可以用父级的外边距替代父级的内边距,但是这种解决方法有可能会对布局造成一定的影响,因此使用时请额外注意。

### 3. 元素设置超出隐藏,不设置固定高度,为何也能够隐藏

前面曾经提到过超出隐藏对于元素固定范围的要求。

在标签没有设置固定高度的情况下,也可以使用 overflow 配合 margin 的负值,实现超出内容的隐藏。

以如下代码为例,在使用 margin 负值与 overflow 之前,内容区的高度为 $(50+2)\times 4=208\text{px}$ 。由于没有为父级 div 设置高度,因此它的高度应当由内容撑开,也就是 208px。

此时,内容是 208px,父级元素大小是 208px,为父级元素设置 overflow: hidden;之后没有任何的反应和效果。

之后,为子元素设置底部外边距,为-2 像素,那么每个子元素的实际高度为 $(50+2)-2=50\text{px}$ ,父级元素高度为 $200\text{px}$ ,每个子元素的 2 像素虚线边框会与下一个元素重叠起来,最后一个子元素的 2 像素边框会与父级元素的实线边框重叠起来,当再为父级元素设置 `overflow: hidden` 时,父级元素之外的部分就被隐藏掉了。

代码实例：

```
<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - margin 负值配合 overflow</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        .con {
            /* overflow: hidden; */
```





```
        border: 5px solid black;
    }
    .con div {
        /* margin-bottom: -2px; */
        height: 50px;
        border-bottom: 2px dashed red;
    }
</style>
</head>
<body>
    <div class="con">
        <div>文章标题 1</div>
        <div>文章标题 2</div>
        <div>文章标题 3</div>
        <div>文章标题 4</div>
    </div>
</body>
</html>
```

#### 代码解析：

当前代码为设置 margin 负值与 overflow 之前，去掉代码中的注释，即为设置 margin 负值与 overflow: hidden; 之后的实例代码。

#### 4. margin 负值的原理解析

核心原理：为子元素设置 margin 负值；为父级元素设置超出隐藏。

嵌套层数：为了便于控制，通常会多嵌套一层标签；涉及横向浮动元素超出时，必须多嵌套一层结构（在问题区当中能够看到具体实例）。

方法优势：能够让所有的同种标签样式统一，并不会产生额外的样式设置，兼容性好，代码扩展性强。

注意：父级的 padding 可能会影响显示效果，在适当的时候可以多嵌套一层，或者将父级的 padding 值更换为子级元素的 margin 值。

### 6.8.5 margin 负值的问题区

#### (1) margin-bottom 负值能否实现需求？

如果通过顶部边框来实现虚线效果，只需要使用 margin-top 的负值配合 overflow，就能够实现基本需求。如果希望通过 dl 元素底部边框来实现虚线效果，也是能够实现的，需要注意的是，不再使用 margin-top 的负值，而是 margin-bottom 的负值。

#### (2) 如果项目只需要支持现代浏览器，能否直接使用伪类选择器来进行处理和操作？

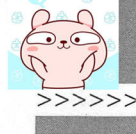
可以。在进行网站开发时，首先要明确网站的兼容要求，然后根据网站的兼容要求，进行功能的选择（例如选择器、CSS3 技术），如果网站要求兼容现代主流浏览器（谷歌、火狐、IE9+），那么完全可以使用 first-child 等伪类选择器来处理背景图。

#### (3) margin 负值的常用场景？

margin 负值主要应用于“同类同级元素，原本应当是统一样式，但是由于布局需要，某







一个或某几个的样式与其他元素样式有细微差别”的场景。

在此给出 margin 负值的常见场景。

场景 1,如图 6.24 所示。

文章列表,  $N$  条数据,  $N-1$  条边框。

此处功能实现可以采用两层结构或三层结构,如果采用两层结构,外层为容器,内层为具体列表项,之后针对每个列表项进行 margin 负值的设置;如果采用三层结构,外层为显示区域容器,中层为具体内容容器,内层为具体列表项,之后为中层容器进行 margin 负值的设置。

场景 2,如图 6.25 所示。

水平方向 margin 负值的应用。

此处功能实现可以采用两层结构或三层结构。

此处的使用方法与场景 1 相同。



图 6.24 margin 负值的应用场景 1  
——文章列表

导航标题1	导航标题2	导航标题3	导航标题4	导航标题5
-------	-------	-------	-------	-------

图 6.25 margin 负值的应用场景 2——水平导航

场景 3,如图 6.26 所示。

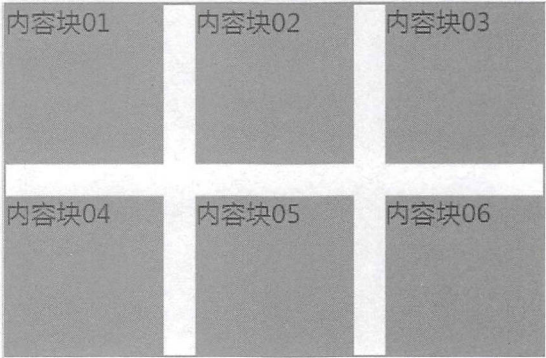


图 6.26 margin 负值的应用场景 3——多个类似布局块

此处功能实现必须使用三层结构,最外层的容器与最内层的 6 个内容块之间,还有一层容器,这个中层容器的宽度需要能够包含下每行的三个内容块。此处不能使用两层结构,原因在于内容块发生了浮动,浮动后元素的位置是根据父级容器剩余空间而自动进行调整的,如果希望在一行当中显示三个内容块,并且每个内容块右侧有一定的空隙,那么父级元素的宽度就必须能够大于等于这个大小。

为最外层的 div 设定固定宽高(外部边框区域)以及 overflow: hidden,内部的每个内容块设定固定的右侧外边距和顶部外边距,之后为中间的 div 设置宽度和 margin-top 负值。





## 6.9 标签选择实战(4)——完成开发

### 6.9.1 考虑超出和 margin 负值

在开发时,需要注意,要针对 `img` 标签设置宽高,为 `a` 标签调整可触区,为 `h2`、`p` 等标签设置超出显示为省略号或超出隐藏,要使用 `margin` 负值完成  $n$  个列表项, $n-1$  条分隔线的效果。

### 6.9.2 完整版代码

代码实例(考虑代码篇幅问题,只保留了两个 `dl`):

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 标签选择完成版案例</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    /* 整体布局 */
    .arc-list {
      width: 676px;
      padding: 14px 50px;
    }
    .arc-con {
      overflow: hidden;
    }
    .arc-list dl {
      margin-top: -1px;
      border-top: 1px dotted #ccc;
    }
    /* 列表项的模块布局 */
    .arc-list dt {
      float: left;
      width: 152px;
      height: 152px;
      padding: 24px;
    }
    .arc-list dd {
      float: right;
      width: 465px;
      height: 150px;
      padding: 25px 0px;
```







```
}
/* 列表项左侧图像 */
.arc-list dt a {
    border: 1px solid #777;
}
.arc-list dt a, .arc-list img {
    display: block;
    width: 100%;
    height: 100%;
}
/* 列表项右侧标题 */
.arc-list h2 {
    margin-bottom: 22px;
}
.arc-list dd a {
    overflow: hidden;
    text-overflow: ellipsis;
    word-break: keep-all;
    white-space: nowrap;
    display: block;
    height: 24px;
    font-size: 18px;
    font-weight: bold;
    color: #333;
}
.arc-list dd a:hover {
    text-decoration: underline;
}
/* 列表项右侧分类与发布时间 */
.arc-list div {
    margin-bottom: 16px;
}
.arc-list span {
    margin-right: 54px;
    color: #777;
}
/* 列表项右侧文章介绍 */
.arc-list p {
    overflow: hidden;
    height: 72px;
    line-height: 24px;
    color: #777;
}
</style>
```







```
</head>
<body>
  <div class="arc-list">
    <div class="arc-con">
      <dl class="clearfix">
        <dt>
          <a href="#" title=""></a>
        </dt>
        <dd>
          <h2><a href="#" title="">经典的斐波那契数列与 arguments 经典的斐
波那契数列与 arguments.callee</a></h2>
          <div>
            <span>2016-08-15</span>
            <span>文章类别: JavaScript</span>
          </div>
          <p>HTML5 学堂: 经典的斐波那契数列与 arguments.callee。提到斐波那契
数列,很多人还不是太清楚,但是如果提到兔子的繁殖这个经典题目,相信学过计算机语言的人们会
感觉很亲切。提到斐波那契数列,很多人还不是太清楚,但是如果提到兔子的繁殖这个经典题目,相
信学过计算机语言的人们会感觉很亲切。</p>
        </dd>
      </dl>
      <!-- dl 部分代码应为多个,此处省略 -->
    </div>
  </div>
</body>
</html>
```

#### 代码解析:

出于显示效果的考虑,此处给出了包含样式的完整版代码,但是请注意本章主要是针对结构标签的选择与处理进行讲解,因此请重点查看结构标签,先理解模块布局当中需要考虑的影响因素,掌握图像、超链接、文本超出等方面的思路和控制方法。

关于代码当中涉及的 line-height、font-size、font-weight、color、text-decoration 几个属性,均为文本等细节类样式,这些样式的具体讲解,请详见第 8 章。

### 6.9.3 总结

谁都没有想到,看上去简简单单的标签选择,在真正开发时竟然需要考虑如此之多的问题。

在网页开发的标签选择方面,并没有一个绝对正确的答案,因为在标签选择时,需要考虑各个方面的因素,不同方法在不同方面有着各自的优劣势,此时作为开发工程师的我们需要找到一个平衡点,挑选出一种相对最优的方法,作为最终“答案”。到目前为止,在进行模块标签选择时,需要注意的方方面面问题就已经讲解完成了,也书写出了相对较好的完成版代码。在此,来回顾一下模块标签选择与构思、调整的流程和思路,如图 6.27 所示。



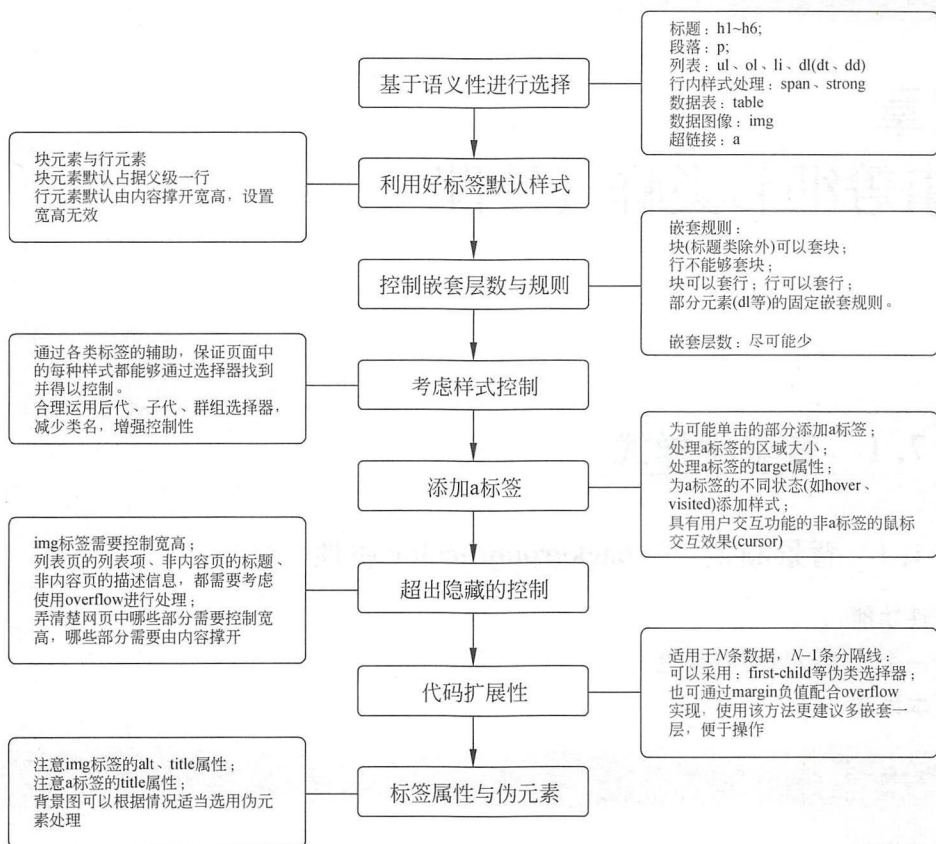


图 6.27 标签选择总结

至此，已完成标签的选择，通过盒模型样式、浮动等显示样式的处理，实现基本模块的布局之后，接下来要进行的的就是下一章当中的文本样式处理了，除了处理标签文本的样式之外，有时还需要结合行高、对齐等文本类属性，对当前已经完成的模块布局进行微调。



# 第7章

## 文本等细节类样式处理



### 7.1 背景类样式

#### 7.1.1 背景颜色——background-color 属性

属性功能：

为一个元素设置背景颜色。

基本语法：

```
background-color: red;
```

代码解析：

设置元素背景颜色为红色。

属性值如表 7.1 所示。

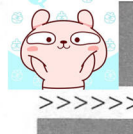
表 7.1 背景颜色的属性值

值	描 述
transparent	默认。背景颜色为透明
color_name	规定颜色值为颜色名称的背景颜色(比如 red)
hex_number	规定颜色值为十六进制值的背景颜色(比如 #ff0000 或 #f00)
rgb_number	规定颜色值为 rgb 代码的背景颜色(比如 rgb(255,0,0))
inherit	规定应该从父元素继承 background-color 属性的设置

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 设置背景颜色</title>
  <link rel = "stylesheet" href = "../css/reset.css">
```





```
<style>
    .box {
        width: 200px;
        height: 200px;
        background: rgb(255, 0, 0);
    }
</style>
</head>
<body>
    <div class = "box"></div>
</body>
</html>
```

颜色可以是一个名称标示的关键字,如 red,这种方法需要记忆比较多的英文单词,所以对于中国人来说,并不是很实用。

颜色也可以使用 6 位十六进制的数字,如 #ffffff; 如果每个参数各自在两位上的数字都相同,那么将每两个数字缩写为一个参数,如: #ff8800 可以缩写为 #f80; #ff8900 则不能够进行缩写。

rgb 和 rgba 也是比较直观表示颜色的一种方式,通过数字来表示一个颜色,rgb 中共有三个数字,每个数字和每个数字之间使用逗号分隔,第一个数字表示红色的色值,第二个数字表示绿色的色值,第三个数字表示蓝色的色值,每个色值的取值范围为 0~255。rgba 的知识请详见 7.2 节。

7.1.2 背景图片——background-image 属性

属性功能:

为一个元素设置背景图像。

基本语法:

```
background-image: url('images/HTML5.jpg');
```

代码解析:

设置元素背景图像为 images 文件夹中的 HTML5.jpg 文件。

注意: 在 url 的括号中书写的是具体的文件路径,这个文件路径可以使用引号(单引号或双引号)包含起来,不使用引号包含路径也是可以实现功能的。无论采取哪种方法均能够被浏览器解读,但请在自己的代码使用当中,保持一致的用法。

属性值如表 7.2 所示。

表 7.2 背景图片的属性值

值	描 述
none	默认值。不显示背景图像
url('URL')	指向图像的路径
inherit	规定应该从父元素继承 background-image 属性的设置

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 设置背景图像</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      width: 400px;
      height: 400px;
      background-image: url('images/HTML5.jpg');
    }
  </style>
</head>
<body>
  <div class = "box"></div>
</body>
</html>
```

7.1.3 背景重复——background-repeat 属性

属性功能：

当为一个元素设置背景图之后,通过 background-repeat 属性来控制图像是否重复以及重复的方式。

基本语法：

```
background-repeat: no-repeat;
```

代码解析：

设置元素背景图像的重复方式为“不重复”。

**注意：**背景重复属性针对背景图有效,而针对背景颜色不生效。

属性值如表 7.3 所示。

表 7.3 背景重复的属性值

值	描 述
repeat	默认。背景图像将在垂直方向和水平方向重复
repeat-x	背景图像将在水平方向重复
repeat-y	背景图像将在垂直方向重复
no-repeat	背景图像将仅显示一次
inherit	规定应该从父元素继承 background-repeat 属性的设置



背景重复的显示效果如图 7.1 所示。

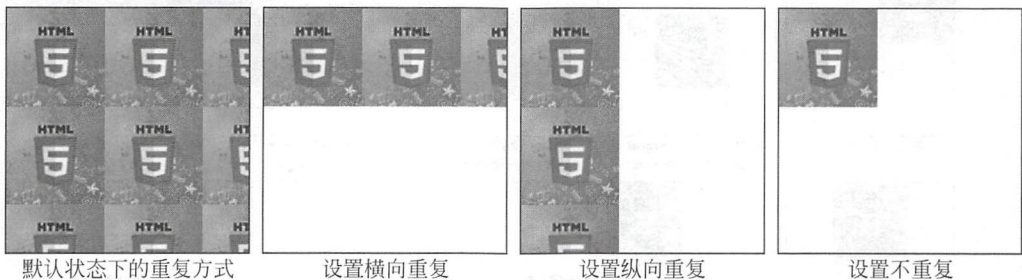


图 7.1 背景重复的显示效果

### 7.1.4 背景定位——background-position 属性

属性功能：

当为一个元素设置背景图之后,通过 background-position 属性来控制图像从哪个位置开始显示。

基本语法：

```
background-position: center center;
```

代码解析：

设置元素背景图像的初始位置为“水平居中垂直居中”。

注意：背景位置属性针对背景图有效,而针对背景颜色不生效。通常会和“背景不重复”一同使用。

属性值如表 7.4 所示。

表 7.4 背景定位的属性值

值	描 述
left   right   center   top   bottom	当设置两个值时,第一个值表示水平方向,第二个值表示垂直方向。 当设置一个值时表示水平方向,此时第二个值默认为“center”
百分比	当设置两个值时,第一个值表示水平方向,第二个值表示垂直方向。 当设置一个值时表示水平方向,此时第二个值默认为“50%”
像素/其他 CSS 单位	当设置两个值时,第一个值表示水平方向,第二个值表示垂直方向。 当设置一个值时表示水平方向,此时第二个值默认为“center”

设置不同背景位置的显示效果如图 7.2 所示。

注：如果使用像素值,水平方向的右边和垂直方向的下边为正值,而水平方向的左边和垂直方向的上边为负值。

### 7.1.5 背景关联——background-attachment 属性

属性功能：

用于设置背景图像是否固定或者随着页面的其余部分滚动。

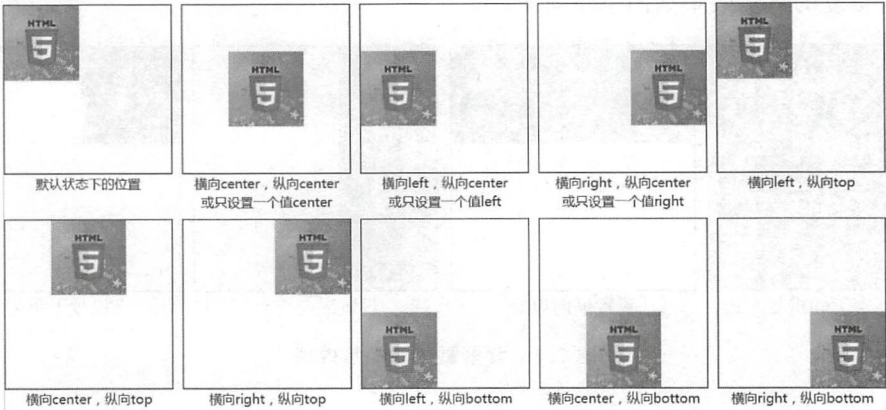


图 7.2 设置不同背景位置的显示效果

该属性生效的基本条件是：为一个元素设置背景图且该元素高度较高(超出了浏览器窗口,出现了滚动条),通常多用于 html 或 body 标签。

基本语法：

```
background-attachment: fixed;
```

代码解析：

设置元素背景图像不随着页面的滚动而运动,固定保持不变。

属性值如表 7.5 所示。

表 7.5 背景关联的属性值

值	描 述
scroll	默认值。背景图像会随着页面其余部分的滚动而移动
fixed	当页面的其余部分滚动时,背景图像不会移动
inherit	规定应该从父元素继承 background-attachment 属性的设置

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 设置背景关联</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    body {
      background-image: url('../images/大图像.jpg');
      background-repeat: no-repeat;
      background-attachment: fixed;
    }
    .box {
      width: 300px;
```



```

        height: 1000px;
        border: 5px solid black;
    }
</style>
</head>
<body>
    <div class = "box"></div>
</body>
</html>
```

### 7.1.6 复合写法——background 属性

属性功能：

将 background 的各类属性设置在一起,用合写的方式进行控制,类似于 border 属性。

基本语法：

```
background: red url('../images/图片.jpg') center center no-repeat fixed;
```

代码解析：

在 background 属性当中,可以同时设置背景颜色、背景图片、背景重复、背景定位、背景关联等属性。

背景实现方法对比如表 7.6 所示。

表 7.6 background 的不同写法对比

方法 1(拆写)	方法 2(合写)
background-image: url('../images/大图像.jpg'); background-repeat: no-repeat; background-attachment: fixed;	background: url('../images/大图像.jpg') no-repeat fixed;

在实际的开发当中,通常并不会分开书写各种背景属性,较多情况下都是使用 background(背景)进行统一控制,合写的方式能够使代码量降到最低。

### 7.1.7 背景类样式的相关问题

(1) 背景颜色和背景图,从哪里开始显示?

解决该方法很简单,为元素设定宽度、高度、背景颜色、内边距值,再设置一个较粗的点线或虚线边框,通过测试来查看背景颜色或背景图是从内容区(width height)、内边距区域、边框区域这三种的哪种位置开始显示的,如图 7.3 所示。

能够看出,背景颜色是从边框区域开始显示的;而背景图是从内边距区域开始显示的。

(2) 背景图和背景颜色同时设置时,是什么效果?

当为一个元素同时设置背景颜色和背景图,且背景图尺寸小于元素区域时,元素会被背景图覆盖,而背景图之外的部分使用背景颜色进行填充。

默认情况下,背景图会横纵向重复,铺满整个元素,为了防止这个问题,需要为背景设置

“no-repeat”，如图 7.4 所示。

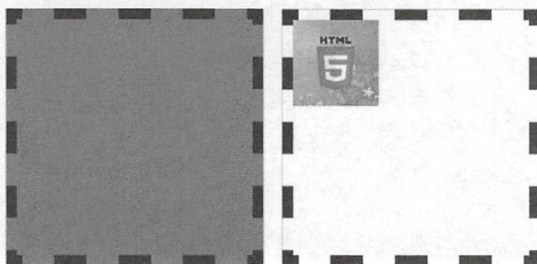


图 7.3 背景颜色和背景图的显示位置

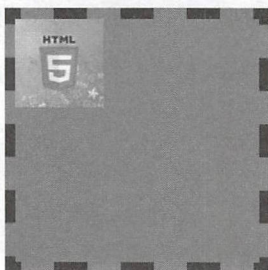


图 7.4 同时设置背景颜色和背景图的显示效果

(3) CSS 文件中，背景图片路径，依照“CSS 文件”还是“引入该文件的 HTML 文件”的位置进行计算？

经过代码测试发现，CSS 文件中背景图的路径，是根据 CSS 文件位置进行计算的，如图 7.5 所示。

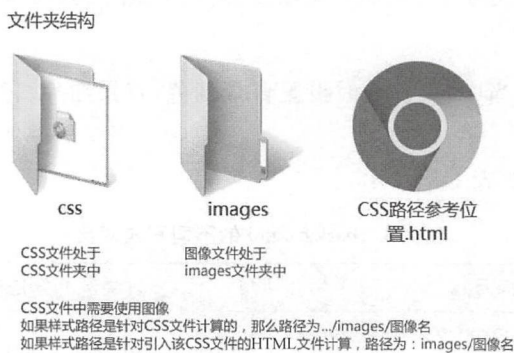


图 7.5 背景图路径的计算方式



## 7.2 透明背景

### 7.2.1 opacity 与 filter

#### 1. filter

最初，CSS 并没有定义透明度的标准属性，不同的浏览器也就定义了自己的专有属性。

IE 浏览器通过设置 CSS 滤镜 (filter 属性) 来实现透明度。书写格式为 filter: alpha (opacity=number)，其中，number 是一个数字，取值范围为 0~100，0 表示 100% 透明度 (即完全透明)，100 表示的是 0% 的透明度 (即完全不透明)。

基本语法：

```
filter: alpha(opacity = 50); /* 表示 50 % 的透明度 */
```



## 2. opacity

火狐浏览器提出了自己的透明度私有属性,使用 `opacity` 属性来实现透明度。书写格式为 `-moz-opacity: number`。其中, `number` 是一个数字,取值范围为 0~1,0 表示 100% 的透明度(即完全透明),1 表示 0% 的透明度(即完全不透明)。IE8-浏览器不支持。

基本语法:

```
-moz-opacity: 0.5;           /* 表示 50 % 的透明度 */
```

## 3. W3C 对 CSS 的修改

随着透明度的广泛应用,W3C 在 CSS 版本中添加了透明度的专有属性 `opacity`,该属性得到了各个主流浏览器以及较高版本 IE 浏览器的支持。书写格式为 `opacity: number`。其中, `number` 是一个数字,取值范围为 0~1,其中,0 表示 100% 的透明度(即完全透明),1 表示 0% 的透明度(即完全不透明)。

基本语法:

```
opacity: 0.5;                /* 表示 50 % 的透明度 */
```

215

## 4. 如何书写兼容所有浏览器的透明度代码

由于当前主流浏览器以及较高版本 IE 浏览器均支持 `opacity`,而低版本浏览器不支持 `opacity`,支持 `filter`,因此,只需要同时设置这两种属性即可。

代码实例:

```
.con {  
    filter: alpha(opacity=50);  
    opacity: 0.5;  
}
```

## 5. opacity 的问题

使用 `opacity` 属性,针对某一个元素设置透明度之后,会引发这个元素内部的子元素跟着变得半透明,简言之就是透明度会影响后代。即便是为后代元素设置透明度为完全不透明,显示方面还是会受到父级元素透明度的影响。

代码实例:

```
<!doctype html>  
<html>  
<head>  
    <meta charset = "UTF - 8">  
    <title>HTML5 布局之路 - opacity 对子级元素的影响</title>  
    <link rel = "stylesheet" href = "../css/reset.css">  
    <style>  
        .box, .noopacity {
```

```

        float: left;
        width: 150px;
        height: 150px;
        margin-right: 10px;
        border: 5px solid black;
    }
    .box {
        opacity: 0.4;
    }
</style>
</head>
<body>
    <div class="box">
        父级元素的文本内容
        <div>子元素的文本内容</div>
    </div>
    <div class="noopacity">
        父级元素的文本内容
        <div>子元素的文本内容</div>
    </div>
</body>
</html>

```

显示效果如图 7.6 所示。

#### 代码解析：

左右两个模块的结构是一样的，只是为左侧的父级元素设置了 `opacity: 0.4`，此时能够发现，变半透明的并不仅是父级元素的边框和文本内容，子级元素的文本内容也变得半透明了。

在默认情况下，子级元素的半透明为 100% 不透明，但是由于它处于半透明的父级元素当中，所以显示效果也受到了影响。

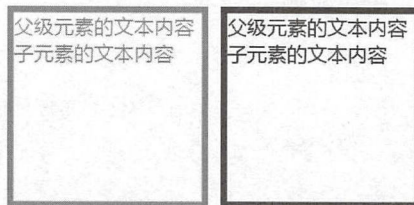


图 7.6 opacity 对子元素透明度的影响

### 6. 如何解决 opacity 带来的影响

在实际开发当中，时常会用到透明度来实现一些效果，opacity 带来的子级透明度问题就成了必须解决的问题，此处需要使用到定位的知识。实现的基本原理是：通过“定位”，将视觉效果模仿成一个元素嵌套另一个元素的样式，但这两个元素实际上是同级关系。对于定位知识，在第 8 章当中会做详细讲解。

### 7.2.2 rgba 控制

rgba 是一个具体属性值，而非属性。可以使用 rgba 定义背景颜色 (background-color)，也可以定义文本颜色 (color)。IE8-浏览器不支持使用 rgba 模式实现透明度，可使用 IE 滤镜 (filter) 处理。



### 基本语法:

```
背景色/文本色: rgba(rnum, gnum, bnum, alpha);
```

### 代码解析:

rnum、gnum、bnum, 分别表示红、绿、蓝, alpha 表示透明度。前三个数字的取值范围为 0~255 的整数; alpha 的取值范围为 0~1, 可以使用小数, 但是只能精确到两位小数。

### 代码实例:

```
background: rgba(255, 0, 0, 0.6); /* 表示 60% 透明度的红色 */
```

## 7.2.3 transparent

transparent 表示全透明, 得到了 IE9+ 以及各个主流浏览器的支持。可以为背景颜色、边框颜色或文字颜色设置 transparent。

## 7.2.4 透明背景的问题区

### (1) -moz 是什么意思?

-moz 是火狐浏览器的浏览器内核前缀, 在 CSS 代码读取当中, 只有火狐浏览器会读取这个前缀后面的样式代码。关于浏览器内核以及浏览器内核前缀, 在第 13 章当中会进行详细讲解。

### (2) 为何 rgb 的取值范围为 0~255?

红绿蓝, 每一种颜色均是由 8 位的二进制码组成。 $2^8 = 256$ , 也就是说红、绿、蓝三种颜色, 各提供了 256 种, 分别使用 0~255 来表示。



## 7.3 背景图合并

### 7.3.1 什么是背景图合并

背景图合并技术, 是将网站当中需要使用的各种背景图片, 通过 Photoshop 合并到一张图当中, 之后运用 HTML+CSS 技术中的 background-position 属性进行各个背景图图片位置的控制。

### 7.3.2 为何进行背景图合并

不进行图片的合并, 只要开发工程师将文件路径书写正确, 每张背景图片也能够正常加载。之所以要额外增加工作任务, 进行背景图的合并, 主要是为了降低“服务器请求压力”。

#### 1. 服务器请求压力

和外部引入的 CSS 文件类似, 图片也需要由客户端向服务器发出申请, 再由服务器将图片发回。

并非所有的服务器请求都会造成服务器请求压力, 服务器端能够同时处理的任务数量

是一定的,当请求次数过多,超出了服务器能够同时处理的请求数量时,才会造成“服务器请求压力”。

在网站中的各个背景图,通常其大小并不大,每个文件发送的(从服务器端下载到客户端)时间并不多,但是“排队”的时间却有可能很长。当将多张背景图合并成一张或几张图片时,会减少客户端向服务器请求的次数,减轻了服务器的压力,这样能够使网页的加载速度加快,提升用户体验。

## 2. 生活中的“服务器请求压力”

生活中去火车站取火车票,将取票窗口和取票机看作“服务器”,把每位旅客看作“客户端”。取票的过程就如同“客户端”在向“服务器”获取数据,而旅客排队等待的过程就是“服务器”响应“客户端”请求的过程。

假设火车站总共开放了 20 个人工取票窗口,提供了 80 台自动取票机。那么此时,100 名以下数量的旅客去取火车票都不需要进行排队,超过 100 名旅客时,就需要排队了,如果排队等待的人很多,旅客可能就需要花很多时间在等待的过程当中,此时就对“服务器”造成了“请求压力”。

### 7.3.3 背景图合并的核心技术与操作方法

通过 Photoshop 将各个背景图整合到一张大的图片当中,之后通过调整 background-position 的值来让不同的元素显示不同的背景。

在合并背景图时,需要考虑使用背景图的元素的大小以及背景的重复情况,如图 7.7 所示。

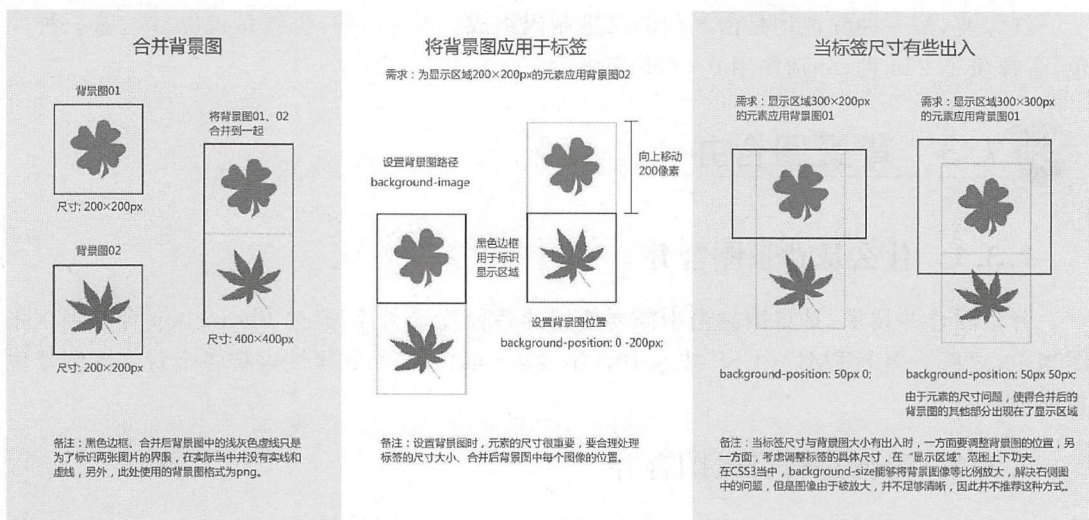


图 7.7 背景图合并的方法

背景图合并的注意事项:

(1) 对于横纵向均需要重复的图像,不要合并到背景图当中。

(2) 需要横向重复或纵向重复的图像,很多并不适合进行背景图合并,要根据具体情况而定。



(3) 各种背景图通常贴近大图左侧或顶端放置,对于“容器水平方向”存在空间的图片,贴左侧进行合并;对于“容器垂直方向”存在空间的图片,贴顶端进行合并。

### 7.3.4 CSS Sprite

#### 1. 什么是 CSS Sprite

CSS Sprite,也有人将其称为 CSS 精灵,是一种网页图片应用处理方式。它允许将一个页面涉及的所有零星图片都合并到一张大图当中,当访问该页面时,载入的图片就不会像以前那样一幅一幅地慢慢显示出来了。

#### 2. CSS Sprite 原理

CSS Sprite 与 Photoshop 的背景图合并一样,就是把网页中一些背景图片整合到一张图片文件中,再利用 CSS 的 background-position 属性进行背景定位。

#### 3. CSS Sprite 使用步骤

(1) 准备需要整合在一起的小图片。

(2) 打开 CSS 背景图合并工具,在线地址: <http://s.uis.cc/pw/>,也可以在本书提供的电子资料当中下载 CSS Sprite 软件。

(3) 选择多张图片,单击“打开”按钮,如图 7.8 所示。

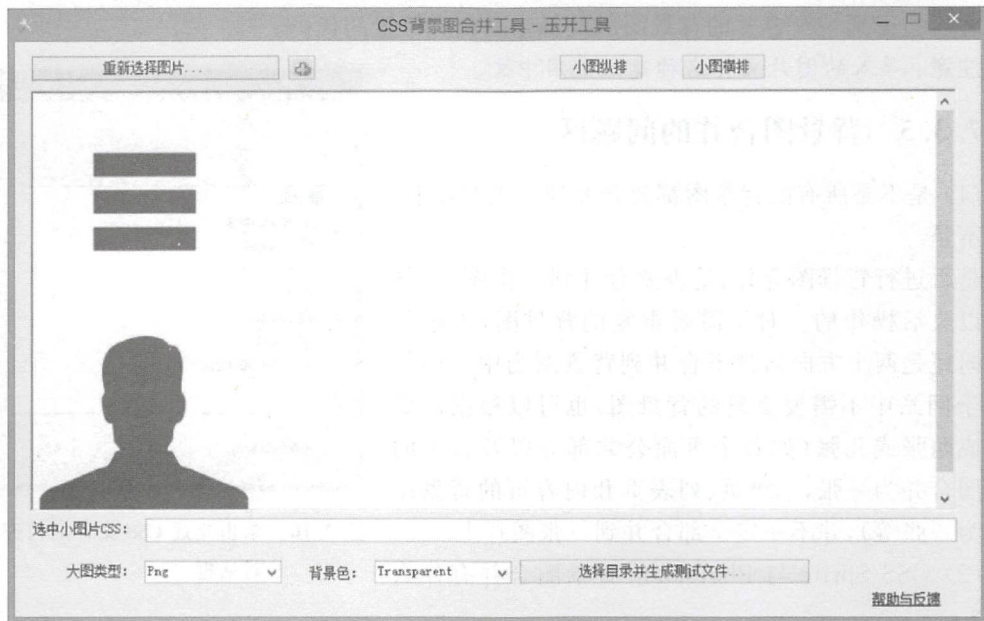


图 7.8 CSS Sprite 操作界面

(4) 排布图片: 可以选择程序提供的按钮完成横竖的默认排布,也可以用鼠标选中图片拖动位置,拖动完成后程序会根据内部图片的位置,生成面积最小的背景图。

(5) 代码生成: 在程序中可以查看每张小图片的 CSS 代码,建议生成图片后再复制生成的代码,如图 7.9 所示。



图 7.9 查看具体每个图片位置的 CSS

(6) 单击“选择目录并生成测试文件”按钮，调整目录，生成图片即可，如图 7.10 所示。生成之后会得到一个合并的背景图以及每张图相应位置的代码。

注意：导入的图片文件名称建议使用中文。

### 7.3.5 背景图合并的问题区

(1) 是不是所有的背景图都要合并到一张图片上？  
不是。

是否进行背景图合并、是否要合并到一张图上，都是可以灵活操作的。对于需要重复的背景图（无论是单方向还是两个方向），都不合并到背景图当中。即便是对于网站中不需要重复的背景图，也可以根据需要进行合并成两张或几张（如各个页面公共部分以及首页的背景图合并为一张，二级页、列表页和内容页的背景图合并为一张等），并不一定全部合并到一张图片上。

(2) CSS Sprite 与 Photoshop 背景图合并有什么区别？

Photoshop 的背景图合并，更多的是人工测算，需要花费的时间较长，需要进行图片位置测算和记录。对于 CSS Sprite 来说，可以直接生成出具体的代码，相对来说操作要更加简单。

从维护角度来说，Photoshop 要比 CSS Sprite 更容易维护，当页面背景出现少许改动时，可以任意调整图片，但 CSS Sprite 需要对原有合并好的整张图片进行修改，灵活度不够。

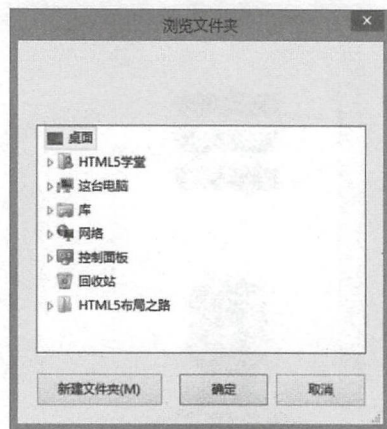


图 7.10 单击生成 CSS 文件后打开的对话框





## 7.4 段落样式

### 7.4.1 line-height

属性功能：

设置元素当中每行文本的行高(行间距)。

基本语法：

```
line-height: 20px;
```

代码解析：

设置元素的文本行高为 20 像素大小。

属性值如表 7.7 所示。

表 7.7 line-height 的属性值

值	描 述
normal	默认。设置合理的行间距
number	设置数字,此数字会与当前的字体尺寸相乘来设置行间距
length	设置固定的行间距。由浮点数字和单位标识符组成的长度值,允许为负值。设置数值越大,文本段落中间的行距越大
%	基于当前字体尺寸的百分比行间距
inherit	规定应该从父元素继承 line-height 属性的值

当 line-height 属性取值小于字体大小时,多行文本情况下,会发生上下行文本重叠的现象; line-height 不能够为负值,如果将行高设置为负数,浏览器会按照默认行高来渲染段落文本。

在默认情况下,浏览器字体大小为 16 像素,那么 IE 的默认行高即为 19 像素,谷歌下默认行高为 21 像素,火狐下的默认行高为 22 像素。

对于单行文本,行高有着“垂直居中”的特殊应用。可以将文本的 line-height 设置成元素的高度值大小,当文本行高等于元素高度时,这段文本就在元素当中垂直居中显示。对于多行文本,这种方法无效。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - line-height</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .test {
      width: 300px;
      height: 40px;
```

```
margin-bottom: 20px;
border: 1px solid red;
line-height: 40px;
}
</style>
</head>
<body>
  <div class = "test">单行文本在元素中垂直居中</div>
  <div class = "test">当文本内容为多行时,每行文本均为设定的行高,就会超出元素显示范围,造成显示上的问题</div>
</body>
</html>
```

显示效果如图 7.11 所示。

7.4.2 text-decoration

属性功能：  
设置元素的文本修饰，如下画线、上画线等。  
基本语法：

```
text-decoration: underline;
```

代码解析：  
设置元素的文本下画线。  
属性值如表 7.8 所示。

单行文本在元素中垂直居中

当文本内容为多行时，每行文本均为设定

的行高，就会超出元素显示范围，造成显示上的问题

图 7.11 line-height 实现纵向居中的特殊用法

表 7.8 text-decoration 的属性值

值	描 述
none	默认。定义标准的文本
underline	定义文本下的一条线(下画线)
overline	定义文本上的一条线(上画线)
line-through	定义穿过文本的一条线(删除线)
blink	定义闪烁的文本
inherit	规定应该从父元素继承 text-decoration 属性的值

7.4.3 text-indent

属性功能：  
设置元素的首行文本的缩进。  
基本语法：

```
text-indent: 2em;
```



代码解析：

设置元素的首行文本缩进为两个字符长度。

属性值如表 7.9 所示。

表 7.9 text-indent 的属性值

值	描 述
length	定义固定的缩进。默认值：0
%	定义基于父元素宽度的百分比的缩进
inherit	规定应该从父元素继承 text-indent 属性的值

**注意：**text-indent 可以为负值,负值时表现为“悬挂缩进”。最为常用的单位是 em,em 指文字的倍数。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - text - indent</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box1, .box2 {
      width: 200px;
      border: 1px solid red;
    }
    .box1 p {
      text - indent: 2em;
    }
    .box2 p {
      margin - left: 2em;
      text - indent: - 2em;
    }
  </style>
</head>
<body>
  <div class = "box1">
    <p>HTML5 布局之路,谁说技术不能通俗易懂?HTML5 布局之路,谁说技术不能通俗易懂?
  </p>
  </div>
  <div class = "box2">
    <p>HTML5 布局之路,谁说技术不能通俗易懂?HTML5 布局之路,谁说技术不能通俗易懂?
  </p>
  </div>
</body>
</html>
```

显示效果如图 7.12 所示。

代码解析：

第一个 div 当中的 p 元素,设置了 text-indent: 2em;代表段落文本的首行内容缩进两

个字符。

第二个 div 当中的 p 元素,设置了 text-indent: -2em;代表段落文本的首行内容向前移动两个字符,当字符向前移动时,就会超出文本的区域,所以再使用 margin-left: 2em;将 p 的内容整体向右移动。

### 7.4.4 text-align

属性功能:

设置元素中文本的水平对齐方式。

基本语法:

```
text-align: center;
```

代码解析:

设置元素的文本内容水平居中对齐。

属性值如表 7.10 所示。

表 7.10 text-align 的属性值

值	描 述
left	把文本排列到左边。默认值: 由浏览器决定
right	把文本排列到右边
center	把文本排列到中间
justify	实现两端对齐文本效果
inherit	规定应该从父元素继承 text-align 属性的值

关于 text-align: justify 可以使文本的两端都对齐。在两端对齐文本中,文本行的左右两端都放在父元素的内边界上。然后,调整单词和字母间的间隔,使各行的长度恰好相等。在打印领域很常见两端对齐文本。但是请注意,在最后一行依旧遵循默认对齐方式。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - text-align</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .normal {
      width: 300px;
      border: 1px solid red;
      text-align: left;
    }
    .justify {
      width: 300px;
      border: 1px solid red;
```

HTML5布局之路, 谁说  
技术不能通俗易懂? HTML5  
布局之路, 谁说技术不能  
通俗易懂?  
HTML5布局之路, 谁说技术  
不能通俗易懂? HTML5  
布局之路, 谁说技术不  
能通俗易懂?

图 7.12 首行缩进与悬挂缩进

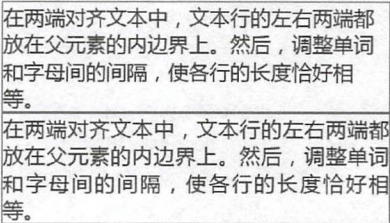


```
        text-align: justify;
    }
</style>
</head>
<body>
    <div class="normal">在两端对齐文本中,文本行的左右两端都放在父元素的内边界上。然后,调整单词和字母间的间隔,使各行的长度恰好相等。</div>
    <div class="justify">在两端对齐文本中,文本行的左右两端都放在父元素的内边界上。然后,调整单词和字母间的间隔,使各行的长度恰好相等。<div>
</body>
</html>
```

显示效果如图 7.13 所示。

7.4.5 vertical-align

属性功能：  
设置元素的垂直对齐方式。  
基本语法：



```
vertical-align: middle;
```

图 7.13 文本两端对齐

代码解析：  
设置元素的垂直居中对齐。  
属性值如表 7.11 所示。

表 7.11 vertical-align 的属性值

值	描 述
baseline	默认。元素放置在父元素的基线上
sub	垂直对齐文本的下标
super	垂直对齐文本的上标
top	把元素的顶端与行中最高元素的顶端对齐
text-top	把元素的顶端与父元素字体的顶端对齐
middle	【较常用】把此元素放置在父元素的中部
bottom	把元素的顶端与行中最低的元素的顶端对齐
text-bottom	把元素的底端与父元素字体的底端对齐
length	以一个数值来排列此元素
%	使用 line-height 属性的百分比值来排列此元素。允许使用负值
inherit	规定应该从父元素继承 vertical-align 属性的值

vertical-align 属性,除了表格类元素支持的比较完善,其他标签支持的并不是太好,因此不建议使用此属性,仅作了解。

代码实例：

```
<!doctype html>
<html>
```

```
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - vertical-align</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .common {
      width: 300px;
      height: 100px;
      margin-bottom: 5px;
      border: 1px solid red;
      vertical-align: middle;
    }
    .change {
      display: table-cell;
    }
    .common p, .data p {
      height: 40px;
      background: #ccc;
    }
    .data {
      margin-bottom: 5px;
    }
    .data td {
      width: 300px;
      height: 100px;
      border: 1px solid red;
      vertical-align: middle;
    }
  </style>
</head>
<body>
  <div class = "common">
    <p>div 中的段落</p>
  </div>
  <table class = "data">
    <tbody>
      <tr>
        <td>
          <p>td 中的具体段落</p>
        </td>
      </tr>
    </tbody>
  </table>
  <div class = "common change">
    <p>改变展示类型的 div 中的段落</p>
  </div>
</body>
</html>
```



显示效果如图 7.14 所示。

代码解析：

代码实例当中采取了三种不同的结构，均希望让 p 元素在其父级元素当中垂直居中。第一个父级元素是 div，第二个父级元素是 td，第三个父级元素是修改展示类型的 div 元素（展示类型修改为单元格 table-cell）。

由于表格类元素对 vertical-align 支持较好，其他标签默认情况下不支持，因此第一个 p 元素并没有在 div 当中垂直居中。第三个 p 元素外侧的 div，由于改变了其展示类型，所以拥有了 vertical-align 属性。

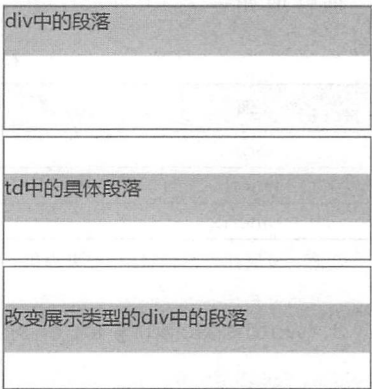


图 7.14 vertical-align 效果

7.4.6 word-spacing 与 letter-spacing

1. word-spacing

属性功能：

设置元素文本的单词间的空白（即字间隔）。

基本语法：

```
word-spacing: 20px;
```

代码解析：

设置元素文本的单词间的空白大小为 20 像素。

属性值如表 7.12 所示。

表 7.12 word-spacing 的属性值

值	描 述
normal	默认。定义单词间的标准空间
length	定义单词间的固定空间
inherit	规定应该从父元素继承 word-spacing 属性的值

注意：该属性可以为负值。如果提供一个正长度值，那么字词之间的间隔就会增加。设置一个负值，会把它拉近。

2. letter-spacing

属性功能：设置元素文本的字符间的空白（字符间距）。

基本语法：

```
letter-spacing: 2px;
```

代码解析：

设置元素文本的字符间空白长度为 2 像素。

属性值如表 7.13 所示。

表 7.13 letter-spacing 的属性值

值	描 述
normal	默认。规定字符间没有额外的空间
length	定义字符间的固定空间(允许使用负值)
inherit	规定应该从父元素继承 letter-spacing 属性的值

注意：该属性可以为负值，如果提供一个正长度值，那么字符之间的间隔就会增加。设置一个负值，会把它拉近。

3. word-spacing 与 letter-spacing 的代码实例

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - word - spacing 与 letter - spacing</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap div {
      height: 50px;
      border: 2px solid black;
      margin - bottom: 5px;
    }
    .word - longer {
      word - spacing: 2px;
    }
    .word - shorter {
      word - spacing: - 2px;
    }
    .letter - longer {
      letter - spacing: 2px;
    }
    .letter - shorter {
      letter - spacing: - 2px;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div>正常模式: HTML5 布局之路 - The road of HTML5 layout.</div>
    <div class = "word - longer">加大 word - spacing: HTML5 布局之路 - The road of HTML5
    layout.</div>
    <div class = "word - shorter">缩小 word - spacing: HTML5 布局之路 - The road of HTML5
    layout.</div>
    <div class = "letter - longer">加大 letter - spacing: HTML5 布局之路 - The road of
    HTML5 layout.</div>
    <div class = "letter - shorter">缩小 letter - spacing: HTML5 布局之路 - The road of
    HTML5 layout.</div>
  </div>
</body>
</html>
```



显示效果如图 7.15 所示。

正常模式：HTML5布局之路 - The road of HTML5 layout.
加大word-spacing：HTML5布局之路 - The road of HTML5 layout.
缩小word-spacing：HTML5布局之路 - The road of HTML5 layout.
加大letter-spacing：HTML5布局之路 - The road of HTML5 layout.
缩小letter-spacing：HTML5布局之路 - The road of HTML5 layout.

图 7.15 letter-spacing、word-spacing 设置后的效果

代码解析：

letter-spacing 和 word-spacing 这两个属性都用来添加它们对应的元素中的空白。letter-spacing 添加字母之间的空白，而 word-spacing 添加每个单词之间的空白。word-spacing 对中文无效。

7.4.7 word-wrap 与 word-break

1. word-wrap

属性功能：设置长单词能否换行。

基本语法：

```
word-wrap: break-word;
```

代码解析：

设置文本中的长单词可以换行。  
属性值如表 7.14 所示。

表 7.14 word-wrap 的属性值

值	描 述
normal	只在允许的断字点换行(浏览器保持默认处理)
break-word	在长单词或 URL 地址内部进行换行

2. word-break

属性功能：设置自动换行的处理方法。

基本语法：

```
word-break: break-all;
```

代码解析：  
设置允许在文本的单词内换行。  
属性值如表 7.15 所示。

表 7.15 word-break 的属性值

值	描 述
normal	使用浏览器默认的换行规则
break-all	允许在单词内换行
keep-all	只能在半角空格或连字符处换行

3. word-wrap 与 word-break 的代码实例

代码实例：

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - word - wrap&word - break</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .common {
      width: 300px;
      margin - bottom: 5px;
      border: 1px solid red;
    }
    .word - b {
      word - break: break - all;
    }
    .word - w {
      word - wrap: break - word;
    }
  </style>
</head>
<body>
  <div class = "common">正常的显示效果：《HTML5 布局之路》的作者也是 HTML5 学堂的创始人。
  http://www.h5course.com</div>
  <div class = "common">yidachuangdeyingwenzimuhaimeiyoubanfabeiqufenkai</div>
  <div class = "common word - b">word - break: break - all; 截断的显示效果：《HTML5 布局之路》
  的作者也是 HTML5 学堂的创始人。http://www.h5course.com</div>
  <div class = "common word - b">word - break: break - all; yidachuangdeyingwenzimuhaimeiy
  ubanfabeiqufenkai</div>
  <div class = "common word - w">word - wrap: break - word; 截断的显示效果：《HTML5 布局之路》
  的作者也是 HTML5 学堂的创始人。http://www.h5course.com</div>
  <div class = "common word - w">word - wrap: break - word; yidachuangdeyingwenzimuhaimeiy
  ubanfabeiqufenkai</div>
</body>
</html>
```

显示效果如图 7.16 所示。



正常的显示效果：《HTML5布局之路》的作者也是HTML5学堂的创始人。 <a href="http://www.h5course.com">http://www.h5course.com</a>
word-break: break-all; 截断的显示效果：《HTML5布局之路》的作者也是HTML5学堂的创始人。 <a href="http://www.h5course.com">http://www.h5course.com</a>
word-break: break-all; yidachuangdeyingwenzimuhaimeiyoubanfabeiqufenkai
word-wrap: break-word; 截断的显示效果：《HTML5布局之路》的作者也是HTML5学堂的创始人。 <a href="http://www.h5course.com">http://www.h5course.com</a>
word-wrap: break-word; yidachuangdeyingwenzimuhaimeiyoubanfabeiqufenkai

图 7.16 word-wrap、break-word 设置后的效果

#### 代码解析：

一串英文，没有空格没有其他符号，在浏览器看来是一个整体，不能够随意换行，这也就出现了第二个块中的显示效果。

word-break: break-all;是用来断开单词的，在单词到边界时，下个字母自动到下一行。

使用 word-wrap: break-word;时，是进行强制换行，中文没有任何问题，英文语句也没问题，但是对于长串的英文就不起作用。

### 7.4.8 段落样式的问题区

(1) 为何 text-indent 最常用的单位是 em?

主要是为了方便维护。

假设当前的段落字体大小为 16 像素，如果希望一个个段落的文字能够“首行缩进”两个“格子(字符)”，可以使用 text-indent: 32px，也可以使用 text-indent: 2em。设置之后，假设段落字体大小调整为了 14 像素，如果之前设置缩进时使用的值是“32px”，此时就需要修改为“28px”，如果之前设置缩进时使用的值为“2em”，此时不需要任何变化。

(2) letter-spacing 与 word-spacing 有什么区别?

word-spacing 添加每个单词之间的空白，对中文无效。

letter-spacing 添加字母之间的空白，对于汉字是以一个字进行间隔，对于英文以一个字母进行间隔。



## 7.5 字体类样式

### 7.5.1 color

属性功能：

设置元素的文本颜色。

基本语法：

```
color: #39f;
```

代码解析：

设置元素的文本颜色为色值#3399ff的蓝色。

属性值如表7.16所示。

表 7.16 color 的属性值

值	描 述
color_name	规定颜色值为颜色名称的颜色(比如 red)
hex_number	规定颜色值为十六进制值的颜色(比如 #ff0000)
rgb_number	规定颜色值为 rgb 代码的颜色(比如 rgb(255,0,0))
inherit	规定应该从父元素继承颜色

7.5.2 font-family

属性功能：

规定元素的字体系列。

基本语法：

```
font-family: 'Microsoft YaHei';
```

代码解析：

设置元素的字体为“微软雅黑”。

属性值如表7.17所示。

表 7.17 font-family 的属性值

值	描 述
family-name	用于某个元素的字体族名称或/及类族名称的一个优先表 默认值：取决于浏览器
generic-family	
inherit	规定应该从父元素继承字体系列

相关知识：

- (1) 常用的中文和英文字体：中文页面建议以宋体、微软雅黑为首选，其他字体次之；英文页面建议使用 Arial、Tahoma、sans-serif 等字体。
- (2) 定义字体时，建议采用英文，而非中文，主要是防止某些机器中的字体无法正常解读。常见字体的英文名称如下：黑体-SimHei、宋体-SimSun、微软雅黑-Microsoft YaHei。
- (3) font-family: Arial, 'Microsoft YaHei';这种包含多种字体的设置方法，是为标签设置了多个字体。在设置多个字体时，不同的字体之间使用逗号进行分隔。如果出现“设置多种字体”的情况，浏览器的解读方式为：先查询第一个字体，如果第一个字体当中没有目标文字时，会在第二种字体当中进行寻找，以此类推，如果到最后依旧没有找到，则使用浏览



器的默认字体。对于中英结合的网站,通常需要设置多种字体,此时,英文字体需要书写在中文字体的前面。

7.5.3 font-size

属性功能:

规定元素的字体尺寸大小。

基本语法:

```
font-size: 16px;
```

代码解析:

设置元素的字体为 16 像素大小。

属性值如表 7.18 所示。

表 7.18 font-size 的属性值

值	描 述
length	把 font-size 设置为一个固定的值。可使用 px、em 等单位
%	把 font-size 设置为基于父元素的一个百分比值
xx-small   x-small   small   medium   large   x-large   xx-large	把字体的尺寸设置为不同的尺寸,从 xx-small 到 xx-large。默认值: medium
smaller	把 font-size 设置为比父元素更小的尺寸
larger	把 font-size 设置为比父元素更大的尺寸
inherit	规定应该从父元素继承字体尺寸

相关知识:

- (1) length 和 % 两种属性值,是平时开发当中会使用到的书写方式。
- (2) em: 如果元素的 font-size: 20px,那么  $em = px / 20$ ; 以 em 为单位设置字体大小在移动端开发中使用广泛。
- (3) 要求字体应当为偶数,如果使用奇数字体,在各个浏览器当中都能够使用这个奇数字体进行渲染,如果使用小数字体(如: 20.2px),在 IE8+ 等主流浏览器当中,会保留小数位,同时按照这个数值进行渲染,但是在 IE7-浏览器中,会去掉小数点(不是四舍五入方式,是直接去掉小数点后面的值)。
- (4) 网页中的最小字体: 12px。
- (5) 浏览器的默认字体大小: 16px。

7.5.4 font-style

属性功能:

设置元素的字体风格(是否倾斜)。

基本语法:

```
font-style: italic;
```

代码解析：  
设置元素的字体倾斜。  
属性值如表 7.19 所示。

表 7.19 font-style 的属性值

值	描 述
normal	默认值。浏览器显示一个标准的字体样式
italic	浏览器会显示一个斜体的字体样式
oblique	浏览器会显示一个倾斜的字体样式
inherit	规定应该从父元素继承字体样式

7.5.5 font-weight

属性功能：  
设置元素的文本粗细。  
基本语法：

```
font-weight: bold;
```

代码解析：  
设置元素的文本加粗。  
属性值如表 7.20 所示。

表 7.20 font-weight 的属性值

值	描 述
normal	【常用】默认值。定义标准的字符
bold	【常用】定义粗体字符
bolder	定义更粗的字符
lighter	定义更细的字符
100~900	每 100 为一个单位。定义由粗到细的字符。400 等同于 normal,而 700 等同于 bold
inherit	规定应该从父元素继承字体的粗细

7.5.6 font 复合样式

1. 基本语法  
属性功能：  
简写属性在一个声明中设置所有字体属性。  
基本语法：

```
font: italic bold 12px/20px 'SimSun';
```

代码解析：  
设置元素的字体为宋体、倾斜、加粗,12 像素字体大小,20 像素行高。





相关知识：

(1) font 属性是一个复合属性，能够设置多种字体的属性值，属性值与属性值之间使用空格进行分隔，字体大小与行高之间使用“/”分隔。font 属性当中，必须设置字体大小(font-size)和字体类型(font-family)这两种属性的属性值，否则 font 的代码不会生效。

(2) 在 font 属性当中，可以按如下顺序进行属性值的设置：font-style font-variant font-weight font-size/line-height font-family。

备注：关于 font-variant，作用是将段落设置为小型大写字母，在日常开发当中很少使用，在此不做讲解。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - font 中必不可少的属性</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .con1 {
      font: bold 20px "SimHei";
    }
    .con2 {
      font: italic 24px;
    }
  </style>
</head>
<body>
  <div class = "con1">设置了字体大小和字体类型</div>
  <div class = "con2">没有设置字体类型</div>
</body>
</html>
```

显示效果如图 7.17 所示。

2. 关于 font 合写与分写的选用

在一个网站当中，大部分的字体类型都是一种（如“微软雅黑”），而大部分文本的字体大小也通常是一个固定的像素值。因此，会在重置文件当中设置字体类型和字体大小，这样在具体开发时，只需要为特殊的字体类型和字体大小设置 font-family 和 font-size 即可。

由于 font 的设置必须书写字体类型和字体大小，在真正使用时会增加额外的代码量，也会让代码的书写变得不方便。

因此，在实际开发当中更倾向于分开设置字体的各类属性，而不采用 font 这个复合属性。

设置了字体大小和字体类型  
没有设置字体类型

图 7.17 font 属性设置效果



## 7.5.7 网络字体

### 1. 网络字体的作用

有时候,由于设计的需要,网站中难免会用到一些比较特殊的字体(如“华文行楷”)来装饰网站,让其不那么死板,然而,用户计算机上不一定会有这种字体,也就是说,只有客户端安装了这个字体才能正常显示,否则页面渲染会调用客户端计算机上已有的字体来替代开发工程师定义的字体。因此,实现个性化字体不能单纯使用 font-family。

在网络字体推出之前,制作页面的时候会用图片来代替这些比较特殊的字体,但是这种方式极不利于后期的维护、修改与 SEO。在 @font-face 网络字体推出之后,这个问题得到了明显的改善。IE9+ 以及各个主流浏览器均支持 @font-face 属性。

### 2. @font-face 的用法

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title> HTML5 布局之路 - 特殊字体(自定义字体)</title>
  <style>
    @font - face {
      font - family: 'STXINGKA';
      src: local('STXINGKA'), url('STXINGKA.ttf') format('truetype');
    }
    .con {
      font - family: 'STXINGKA';
      font - size: 44px;
    }
  </style>
</head>
<body>
  <div class = "con">独行冰海</div>
</body>
</html>
```

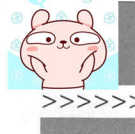
代码解析:

每个字体都有两个 src 定义,第一个是字体的本地名。(注意华文行楷的字体名为“STXINGKA”。)为所有的字体进行这样的设置的作用是:当用户本地已经下载了同样字体的时候,可以节约网上下载的成本。第二个是 url,如果浏览器在本地没有找到这种字体,那么会将 url 中设置的字体加载到页面当中。设置自定义网络字体,必须设置 src 以及 font-family。

使用 src 设置一种格式的字体:

```
src: url('hkshaonv - webfont.eot');
```





使用 src 设置多种格式的字体：

```
src: url('hkshaonv-webfont.eot?#iefix') format('embedded-opentype'),
url('hkshaonv-webfont.woff') format('woff'),
url('hkshaonv-webfont.ttf') format('truetype'),
url('hkshaonv-webfont.svg#HKshaonv') format('svg');
```

3. format

format 是一个可选参数，它的作用是提示该资源 URI 所引用的字体格式，关于字体格式，如表 7.21 所示。

表 7.21 format 属性值与字体格式

format 格式	Font 格式	后缀名
truetype	TrueType	. ttf
opentype	OpenType	. ttf, . oft
truetype-aat	TrueType with Apple Advanced Typography extensions	. ttf
embedded-opentype	Embedded OpenType	. eot
svg	SVG Font	. svg, . svgz

由于各种浏览器对不同字体格式的支持程度有所不同，在书写时就需要设置多种字体格式了，如表 7.22 所示。

表 7.22 浏览器对不同字体的支持程度

浏 览 器	支 持 类 型
IE6,7,8	仅支持 Embedded OpenType(. eot)格式
Firefox 3.5	支持 TrueType、OpenType(. ttf, . otf)格式
Firefox 3.6	支持 TrueType、OpenType(. ttf, . otf)及 WOFF 格式
Chrome, Safari, Opera	支持 TrueType、OpenType(. ttf, . otf)及 SVG Font(. svg)格式

4. 创建自己的字库

国外的一些网站都局部地运用 font-face 来美化网站，但是，和中文字体不同的是，国外的字库只有几十 KB 的大小(26 个大写和 26 个小写英文字母以及一些标点符号)，而中文字库由于存在大量的文字，大部分都是几兆字节甚至几十兆字节不等，为了美化网站而为网站增加几兆字节的流量是很不明智的选择。这时，可以把网页中出现的特殊文字添加到一个新字库里面，把不常用的去掉，做个精简版的字库来满足需求。

可以使用“fontmin”这款工具来创建字体库。

1) 为何推荐 fontmin

与其他的字体生成工具相比较，fontmin 方便、快捷，不需要自己手动抠文字，直接生成各类文字字体，而且代码也可以直接生成。

2) fontmin 的基本特点

既能删除多余文字，又能够调整文字映射。

子集化(只取用当前字体中的部分文字)后的字体删掉了所有没用的空字符，不需要另



开 fontcreator 进行二次精简。

无论原字体的映射怎么乱七八糟,子集化后的映射平台自动改为两个必要的 Unicode 平台,使得在理论上所有的字体都能精简,所有设备上的阅读器都能正常识别。

### 3) fontmin 的下载

fontmin 官网: <http://ecomfe.github.io/fontmin>。

也可以在本书提供的电子资料当中下载(如果希望获取最新版本,请在官方地址当中下载)。

### 4) fontmin 的操作步骤

打开软件(无须安装,即开即用);

输入需要采用特殊字体的文字(注意不要换行);

打开系统中的字体库,选择需要的特殊字体,并将特殊字体拖曳到软件的相应位置,生成;

将生成的文件复制或剪切到开发文件夹的指定位置(会生成各种格式的字体文件以及 CSS 文件,可以使用编辑器打开 CSS 文件)。

### 5) fontmin 使用的注意事项

在使用该工具进行字体处理时,把需要的文字内容粘贴上去,不要换行,否则生成出来的字体,在部分手机当中会出现字符间距增大的现象。

建议采用英文名称命名字体(CSS 文件中 font-family 的属性值及所有的字体名称),而不要采用中文,中文很有可能出现乱码问题。

如果发现在阅读器上还是无法显示,请用 fontcreator 把第一个空字符删了再重新插入这个空字符,就是映射总是 \$ 0000 的那个。这个并不属于软件 bug,fontmin 会完整保留第一个空字符的所有映射,因此可能导致与后来改的字符映射产生冲突,而在移动设备上无法正常显示。按上面的方法手动处理一下就好(并不是每种字体都会出现这个问题,如果出现冲突时会有提示)。

## 7.5.8 字体类样式的问题区

### (1) 如果直接使用特殊字体会不会出现问题?

如果在字体当中直接使用特殊字体,没有提供字体文件,在用户的计算机上可能不能达到想要的效果。这是由于用户的计算机可能是没有这种特殊字体的,在这种情况下可以采用自定义字体的方法。

采用自定义字体方法时,字体文件也不能够直接复制,而需要进行一些处理(取出其中有用的文字,删掉没必要的文字),降低文件大小,以防止网页加载速度过慢。

对于不支持自定义字体的浏览器,需要针对特殊字体切图来实现。

### (2) 当网页中字体小于 12 像素时,会有什么现象出现?

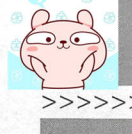
在谷歌浏览器当中,网页字体会自动调整为 12 像素进行显示。如果针对浏览器窗口进行放大时,这些小于 12 像素的字体也会放大(浏览器窗口缩小时没有变化)。

在 IE、火狐浏览器当中,虽然能够识别小于 12 像素的字体,但是实在是太小,可读性差。

代码实例:

```
<!doctype html>
<html>
```





```
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 最小字体的展示效果</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .font12 {
      font - size: 12px;
    }
    .font10 {
      font - size: 10px;
    }
    .font6 {
      font - size: 6px;
    }
  </style>
</head>
<body>
  <div class = "font12"> 12 像素字体：《HTML5 布局之路》</div>
  <div class = "font10"> 10 像素字体：《HTML5 布局之路》</div>
  <div class = "font6"> 16 像素字体：《HTML5 布局之路》</div>
</body>
</html>
```

谷歌浏览器默认显示效果如图 7.18 所示。

IE 浏览器默认显示效果如图 7.19 所示。

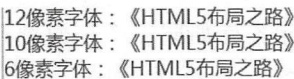


图 7.18 谷歌浏览器小于 12 像素字体的显示效果



图 7.19 IE 浏览器小于 12 像素字体的显示效果

当谷歌浏览器放大时的显示效果如图 7.20 所示。

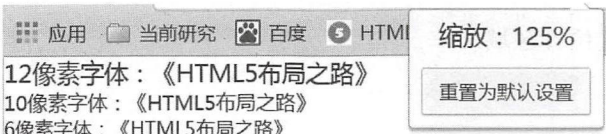


图 7.20 谷歌浏览器放大时小于 12 像素字体的显示效果

(3) italic 与 oblique 有什么区别？

italic 是使用文字的斜体,oblique 是让文字倾斜。通常情况下,italic 和 oblique 文本在 Web 浏览器中看上去是完全一样的,但是对于没有斜体样式的字体,如果使用 italic,则没有效果,需要使用 oblique 属性值来实现倾斜的文字效果。

一种字体在设计时,会有粗体、斜体、下画线、删除线等诸多属性,但是并非所有字体在创建时完整地设计了各种属性和样式,对于一些字体可能只存在正常状态。



## 第8章

# 特殊布局情况——定位布局



### 8.1 定位属性

#### 8.1.1 为何要使用定位

HTML+CSS 布局方式之中,最为常见的有两种布局模式,分别是浮动布局和定位布局。在日常开发当中,使用最多的是浮动布局,当在页面中出现多个元素层叠状态时,会考虑定位布局,例如平时的弹窗、黑色半透明蒙版层、返回顶部、微博等网站顶部的导航条等。

#### 8.1.2 生活中的“定位”——便携贴

网页如同生活中的黑板(或白板),而设置定位的元素就如同一个个便携贴。这些便携贴可以贴在黑板的任意位置,可以在便携贴上再贴便携贴,两个或几个便携贴也可以发生部分区域的覆盖等。

#### 8.1.3 定位——position 属性

属性功能:

为一个元素设置位置区域。

基本语法:

```
position: absolute;
```

代码解析:

设置元素定位方式为绝对定位。

属性值:

position 属性规定了元素的定位类型,所有的元素都可以用 position 来进行定位。position 定位之后的对象将具有块属性。position 属性有 5 个取值,分别为: static、relative、absolute、fixed、inherit,其中,“static”为默认值,可以省略不写,static 元素会忽略任何 top、bottom、left 或 right 声明,如表 8.1 所示。



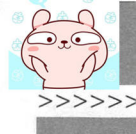


表 8.1 定位的属性值

值	描 述
absolute	生成绝对定位的元素,相对于 static 定位以外的第一个父元素进行定位。 元素的位置通过"left", "top", "right"以及"bottom"属性进行规定
fixed	生成绝对定位的元素,相对于浏览器窗口进行定位。 元素的位置通过"left", "top", "right"以及"bottom"属性进行规定
relative	生成相对定位的元素,相对于其正常位置进行定位
static	默认值。没有定位,元素出现在正常的流中(忽略 top, bottom, left, right 或者 z-index 声明)
inherit	规定应该从父元素继承 position 属性的值

8.1.4 定位对文档流的影响

当对一个元素设置了 position: absolute 和 position: fixed 时,该元素会脱离文档流,从而对父级以及兄弟级元素的布局造成影响。

如果元素设置了 position: relative,根据 W3C 官方文档上来说,并不会脱离文档流,也不会有布局上的问题,但是在实际开发中的情况则是:元素不会脱离文档流,但是可以设置 top、left 等值进行位置的操作。

1. 相对定位的元素不会脱离文档流

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 相对定位不会脱离文档流</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap div {
      width: 200px;
      height: 200px;
      border: 5px solid #f00;
      background: #39f;
    }
    .wrap .box {
      position: relative;
      left: 10px;
      top: 50px;
      background: #3f9;
    }
  </style>
</head>
<body>
  <div class = "wrap">
```



```
<div class = "box">设置了 relative 的 div 元素</div>
<div>HTML5 布局之路</div>
</div>
</body>
</html>
```

显示效果如图 8.1 所示。

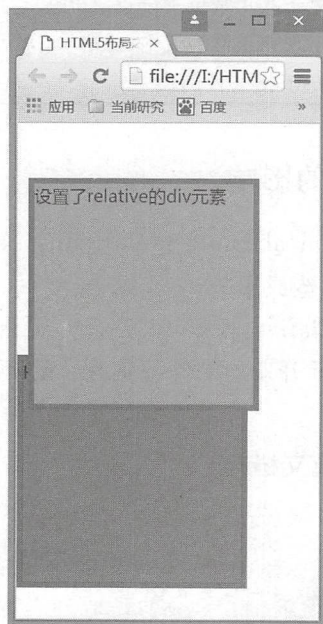


图 8.1 相对定位的元素不会脱离文档流

代码解析：

第一个 div 设置了 `position: relative`；也根据 `top` 等属性值产生了位置偏差，但是第二个 div 的位置并没有受到任何影响，说明相对定位并不会让一个元素脱离文档流而存在。

## 2. 绝对定位的元素会脱离文档流

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 绝对定位会脱离文档流</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap div {
      width: 200px;
      height: 200px;
      border: 5px solid #f00;
      background: #39f;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "box">设置了 absolute 的 div 元素</div>
    <div>HTML5 布局之路</div>
  </div>
</body>
</html>
```





```
    }  
    .wrap .box {  
        position: absolute;  
        left: 10px;  
        top: 50px;  
        background: #3f9;  
    }  
    </style>  
</head>  
<body>  
    <div class = "wrap">  
        <div class = "box">设置了 absolute 的 div 元素</div>  
        <div>HTML5 布局之路</div>  
    </div>  
</body>  
</html>
```

显示效果如图 8.2 所示。

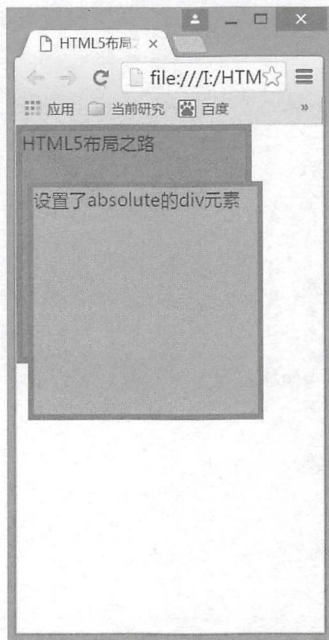


图 8.2 绝对定位的元素会脱离文档流

#### 代码解析：

第一个 div 设置了 `position: absolute`; 第二个 div 的位置受到了影响, 在父级元素中的从左上角开始显示, 似乎父级元素中压根不存在第一个元素一样, 这是由于第一个 div 元素已经脱离文档流。

当元素脱离文档流之后, 就和浮动类似, 既能够为元素设置盒模型属性, 且盒模型属性会生效。但是, `margin` 属性中的 `auto` 会失效。



## 8.2 绝对定位的位置控制

### 8.2.1 设置绝对定位的元素的基本特点

#### 1. 绝对定位元素彼此不互相影响

设置属性值为 `absolute` 会将对象脱离出正常的文档流,进行绝对定位,并不考虑它周围内容的布局。假如其他定位对象已经占据了给定的位置,它们之间不会相互影响,而会在同一位置层叠。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 绝对定位元素可以互相重叠</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      position: relative;
    }
    .box1 {
      position: absolute;
      top: 0px;
      left: 0px;
      width: 150px;
      height: 150px;
      margin: 20px;
      padding: 20px;
      border: 1px solid black;
    }
    .box2 {
      position: absolute;
      top: 0px;
      left: 0px;
      width: 100px;
      height: 100px;
      background: #39f;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "box1">内容块 01 </div>
    <div class = "box2">内容块 02 </div>
  </div>
</body>
</html>
```



显示效果如图 8.3 所示。

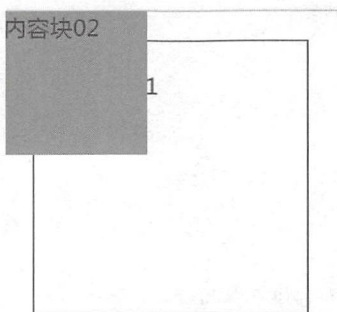


图 8.3 绝对定位元素彼此不互相影响

## 2. 激活绝对定位

要激活对象的绝对定位,必须设置 position 的属性值为 absolute,并且指定 left, right, top, bottom 中的至少一个属性,否则上述属性会使用它们的默认值 auto,这将导致对象遵从正常的 HTML 布局规则,在前一个对象之后立即被呈递。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 激活绝对定位元素</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 200px;
      height: 200px;
      margin: 20px auto;
      border: 5px solid black;
    }
    .box1 {
      position: absolute;
      top: 0px;
      left: 0px;
      width: 150px;
      height: 150px;
      border: 1px solid black;
    }
    .box2 {
      position: absolute;
      top: 0;
      width: 100px;
      height: 100px;
      background: #39f;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "box1">
      <div class = "box2">
      </div>
    </div>
  </div>
</body>
</html>
```

```

        .box3 {
            position: absolute;
            width: 80px;
            height: 80px;
            background: #ccc;
        }
    </style>
</head>
<body>
    <div class = "wrap">
        <div class = "box1">内容块 01 </div>
        <div class = "box2">内容块 02 </div>
        <div class = "box3">内容块 03 </div>
    </div>
</body>
</html>

```

显示效果如图 8.4 所示。

#### 代码解析：

三个绝对定位元素的父级，并没有设置相对定位，因此，三个子元素应当是针对 body 进行定位。

第一个子元素设置了 left 和 top 值，激活了在水平、垂直两个方向上的绝对定位，因此显示于 body 的左上角；

第二个子元素设置了 top 值，仅激活了在垂直方向的绝对定位，因此，垂直方向上显示在 body 的最顶端，但是由于没有设置 left 或 right，水平方向的绝对定位没有被激活，而是“呈递”父级元素（类名为 wrap 的 div）的位置；

第三个元素没有设置 left、top、right、bottom，没有激活绝对定位，因此它默认在原有的文档位置。

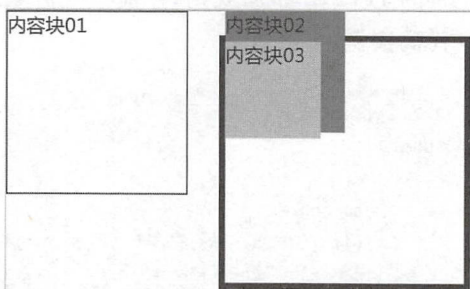


图 8.4 激活绝对定位

### 3. 绝对定位元素针对谁进行定位

如果父级（无限）没有设定 position 属性，那么当前的 absolute 则结合 top, right, left, bottom 属性以浏览器左上角为原始点进行定位。

如果父级（无限）设定 position 属性，且属性值为 relative、absolute、fixed，那么当前的 absolute 则结合 top, right, left, bottom 属性以父级（最近）的左上角为原始点进行定位。

## 8.2.2 定位——left 等属性

### 1. 控制元素偏移的 4 种属性

定位提供了 4 种属性来设置元素的偏移。这 4 种属性分别是：left、right、top 和 bottom。

这 4 种属性均可以设置为负值，单位可以选用绝对单位（如 px），也可以选用相对单位



(如百分比),当使用这 4 种属性进行位置设置时,可以理解为设置“与相对定位元素的某方向位置的距离”。

为父级元素设置相对定位,为子级元素设置绝对定位,之后,针对不同的方向进行设置,显示效果如图 8.5 所示。

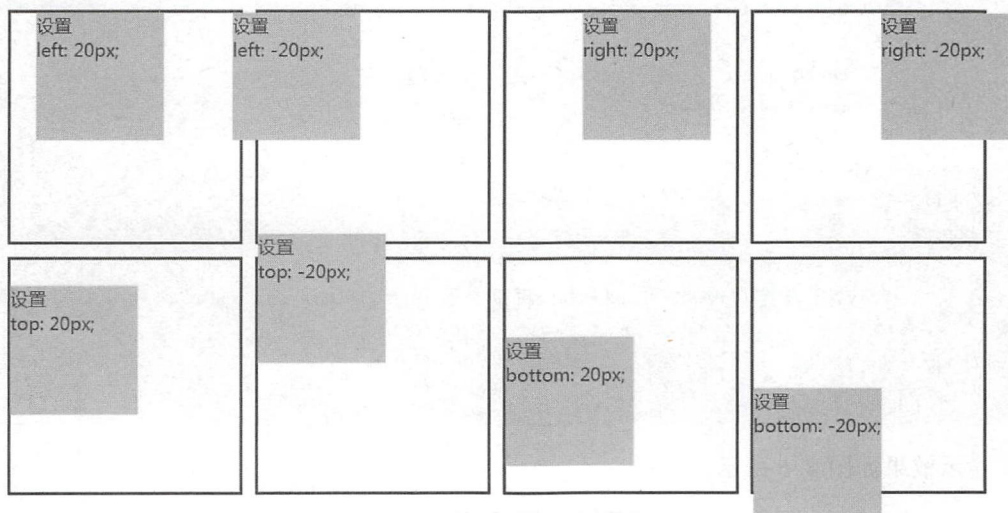


图 8.5 控制定位元素的偏移

## 2. 同时设置同方向的两个值

对于位置控制,可以同时设置水平方向和垂直方向的属性值。例如,设置 left 值为 30px,top 值为 50px。但是,如果同时设置 left、right 的值,浏览器如何解析呢?

对于同时设置水平方向的两个值,或者同时设置垂直方向的两个值的情况,要基于元素是否存在固定宽高而分成两类。

### 1) 绝对定位元素存在固定宽高

当绝对定位元素存在固定的宽(或高)时,在水平方向上,不论先书写 left 值后书写 right 值,还是先书写 right 值后书写 left 值,均按照 left 的值进行渲染;在垂直方向上,不论先书写 top 值后书写 bottom 值,还是先书写 bottom 后书写 top 值,均按照 top 值进行渲染。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 存在固定宽高的 left、right 设置</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      position: relative;
      width: 200px;
      height: 200px;
      border: 2px solid black;
```

```

    }
    .wrap div {
        position: absolute;
        right: 10px;
        left: 10px;
        width: 150px;
        height: 150px;
        background: #ccc;
        font-size: 14px;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div>先设置了 right: 10px;<br>再设置了 left: 10px;</div>
    </div>
</body>
</html>

```

显示效果如图 8.6 所示。



图 8.6 元素存在固定宽度时同时设置 left 和 right

## 2) 绝对定位元素没有设置固定宽高

当绝对定位元素没有设置固定宽高时,合理的 left 与 right(或 top 与 bottom)值都会生效。

元素宽度 = 父级宽度 - left 值 - right 值。

代码实例:

```

<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 无固定宽高的 left、right 设置</title>
    <link rel="stylesheet" href="../css/reset.css">
    <style>
        .wrap {
            position: relative;
            width: 200px;

```



```

        height: 200px;
        border: 2px solid black;
    }
    .wrap div {
        position: absolute;
        left: 10px;
        right: 10px;
        height: 150px;
        background: #ccc;
        font-size: 14px;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div>
            <p>父级元素宽度为 200px;</p>
            <p>不为定位元素设置宽度</p>
            <p>设置 right: 10px;left: 10px;</p>
            <p>元素宽度自动变为 180px</p>
        </div>
    </div>
</body>
</html>

```

显示效果如图 8.7 所示。

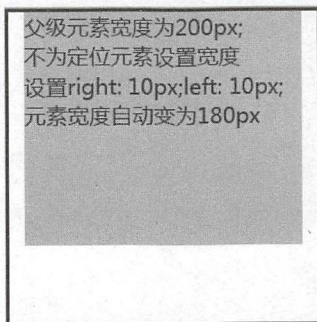


图 8.7 没有固定宽度时,同时设置 left 和 right

元素宽度 = 父级宽度 200px - left 值 10px - right 值 10px = 180px。

如果将 right 设置成 -10px, 则元素宽度 = 200 - 10 - (-10) = 200px。

需要注意的是, 计算得到的元素宽度为负值时, 会自动变为 0 像素。



## 8.3 层级覆盖关系

### 8.3.1 定位——z-index 属性

在一个网页当中, 可能存在着多个定位的元素, 当这些元素互相重叠时, 就涉及层叠的

关系,因此,CSS 当中规定了 z-index 这个属性,来为定位元素设定“层叠级别(或称为堆叠顺序)”,相对定位、固定定位、绝对定位的元素,均可设置 z-index 属性。

定位元素默认的 z-index 值为 0;数字越大,级别越高,数字越小,级别越低;如果将 z-index 值设置为负值,那么层叠顺序会在 HTML 文档之后,如果 HTML 文档设置了背景颜色,这个定位元素将不会被看到。

在定位元素设置 z-index 值之前,层级关系是一样的,默认 z-index 值为 0,但是网页的读取顺序是自上而下的,因此,针对同级别元素来说,后读取的元素级别要高于先读取的元素级别。如果为同一级别下,不同的定位元素设置一样的 z-index 值,那么结果也是后者覆盖前者。如果设置了 z-index,就比较同级别定位元素的 z-index 值大小即可,z-index 值较大者在上面。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 层叠关系</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      position: relative;
    }
    .box1 {
      position: absolute;
      top: 50px;
      left: 50px;
      width: 140px;
      height: 140px;
      background: #ccc;
      border: 5px dashed red;
    }
    .box2 {
      position: absolute;
      top: 100px;
      left: 100px;
      width: 140px;
      height: 140px;
      background: #999;
      border: 5px dotted black;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "box1"></div>
    <div class = "box2"></div>
  </div>
</body>
</html>
```



显示效果如图 8.8 所示。

父级 div 设置了相对定位,两个子元素都设置了绝对定位,在不设置任何位置属性时,两个元素默认从父级元素的左上角开始显示,这两种元素重叠在了一起。为了便于查看,为第一个子元素设置 5 像素的红色虚线边框,浅灰色背景。第二个子元素设置 5 像素的点线边框,深灰色背景。

能够从效果当中查看出,第二个子元素覆盖在第一个子元素的上面。

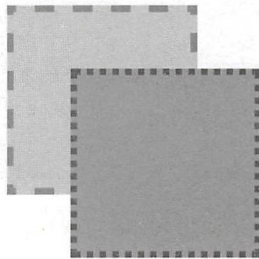


图 8.8 z-index 控制层叠关系

### 8.3.2 多级别的层叠关系比较

z-index 只能针对同级的标签有效,也就是说子标签的 z-index 值对于父标签是无效的,因为两者的级别不同,z-index 是无法比较的。因此,针对两个设置绝对定位的元素进行层级比较时,首先应当比较其父级,再比较子级。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 涉及多级别的定位元素层叠关系</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    .box1, .box2 {
      position: relative;
      width: 200px;
      height: 200px;
      margin-bottom: 10px;
      border: 5px solid black;
    }
    .box1 {
      z-index: 9;
    }
    .box2 {
      z-index: 1;
    }
    .box1 div {
      position: absolute;
      bottom: -50px;
      width: 140px;
      height: 140px;
      background: #ccc;
      border: 5px dashed red;
      z-index: 1;
    }
    .box2 div {
      position: absolute;
```

```
        top: -100px;
        width: 140px;
        height: 140px;
        background: #999;
        border: 5px dotted black;
        z-index: 999;
    }
</style>
</head>
<body>
    <div class="box1">
        <div>第一个 box 中的绝对定位元素</div>
    </div>
    <div class="box2">
        <div>第二个 box 中的绝对定位元素</div>
    </div>
</body>
</html>
```

范例当中,z-index 值设置如表 8.2 所示。

表 8.2 实例中各元素的 z-index 值

具体元素	z-index 值	具体元素	z-index 值
第一个父级元素	z-index: 9;	第一个父级元素中的子元素	z-index: 1;
第二个父级元素	z-index: 1;	第二个父级元素中的子元素	z-index: 999;

由于第一个父级元素的 z-index 值要大于第二个父级元素,所以,第一个子级元素会覆盖第二个子级元素,最终的显示效果如图 8.9 所示。

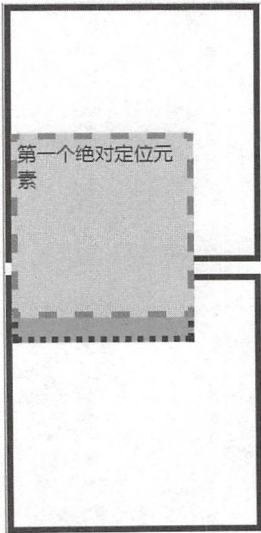


图 8.9 多级别的层叠关系比较





## 8.4 固定定位

fixed 可定位相对于浏览器窗口的指定坐标。这个元素的位置可通过 left、right、top、bottom 属性来规定。不论窗口滚动与否,元素都会留在指定位置,该定位方法在移动端极为常用,在 PC 端有时也会使用于头部导航区,无论内容区的高度有多少,滚动条是否滚动,固定定位的元素始终显示在初始位置。除了 IE6 版本的浏览器之外,当前现代浏览器都可以完美地支持此特性。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 固定定位</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 400px;
      margin: 0 auto;
      padding - top: 40px;
    }
    .nav {
      position: fixed;
      top: 0;
      width: 400px;
      height: 40px;
      background: #39f;
      text - align: center;
      line - height: 40px;
      color: white;
    }
    .box {
      height: 150px;
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "nav">被固定定位的导航条</div>
    <div class = "box">内容块 01</div>
    <div class = "box">内容块 02</div>
    <div class = "box">内容块 03</div>
    <div class = "box">内容块 04</div>
    <div class = "box">内容块 05</div>
  </div>
</body>
</html>
```

显示效果如图 8.10 所示。

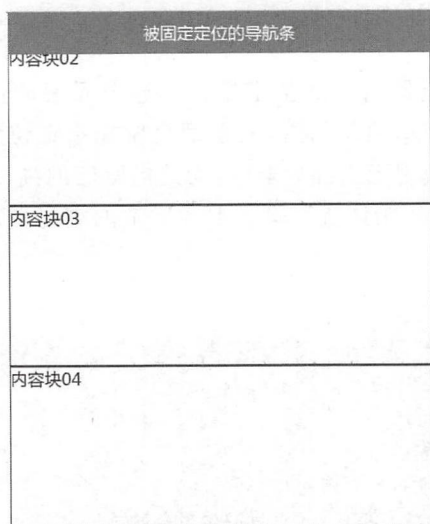


图 8.10 固定定位

#### 代码解析：

当滚动条滚动时，导航条始终显示在浏览器窗口的最顶端。

此处代码当中，有以下两个小细节需要注意。

(1) 对于导航条，设置固定定位之后，如果不设置 top、left、right、bottom，则按照元素默认位置进行显示，如果设置 left 或 right、top 或 bottom，则按照浏览器窗口进行计算。此处希望它能够处于父级元素的位置，因此，并没有为其设置 left(或 right)值。

(2) 由于导航条进行了固定定位，已经脱离文档流，因此它之后的兄弟级元素在计算布局时会忽略掉导航条，但是导航条依旧占据着空间，并且位于所有的结构“之上”，这就会造成内容的遮挡，之后，通过为父元素设置一个 padding-top 值，并且让这个值等于导航条的高度，从而避免这个问题。



# 第9章

## 特殊布局情况

### ——界限控制与伪元素的妙用



#### 9.1 未设置固定宽高元素的界限控制

##### 9.1.1 何处需要考虑界限控制

对于一个元素,如果设置了固定的宽度和高度(width 和 height),其实就意味着为这个元素设置了界限,这种界限不是在这里要考虑的。在这里只考虑“没有设定固定宽高的元素”的界限控制。

如果不为一个块元素设置固定宽度,则该元素在水平方向上默认占满父级的百分之百。如果不为其设置固定高度,则默认由内容撑开高度。对于这两种场景,默认情况下是没有任何问题的,但是如果在某种特殊情形下,就有可能出现一些问题。

##### 9.1.2 最小高度

###### 1. 场景解析

文章列表页,需要呈现一系列的文章,每篇文章的高度设定为 100 像素,列表页最多显示 8 篇文章,由于网站中相关的文章有可能是小于 8 篇的,所以列表高度并不能设定死,需要由内容撑开高度,如图 9.1 所示。但是,当文章数量过少时,也可能会出现一些问题。

###### 问题解析:

如果设置固定高度,内容少的时候会有大量的留白。如果不设置高度,单纯由内容撑开高度,当内容少的时候,这个区域的高度太小,完全无法占满整个浏览器区域,实在太难看。

###### 解决办法:

为元素设置最小高度(min-height),最小高度的值需要根据情况而定。

###### 2. 最小高度的用法

###### 基本语法:

```
min-height: 600px;  
/* 为元素设置最小高度为 600 像素 */
```

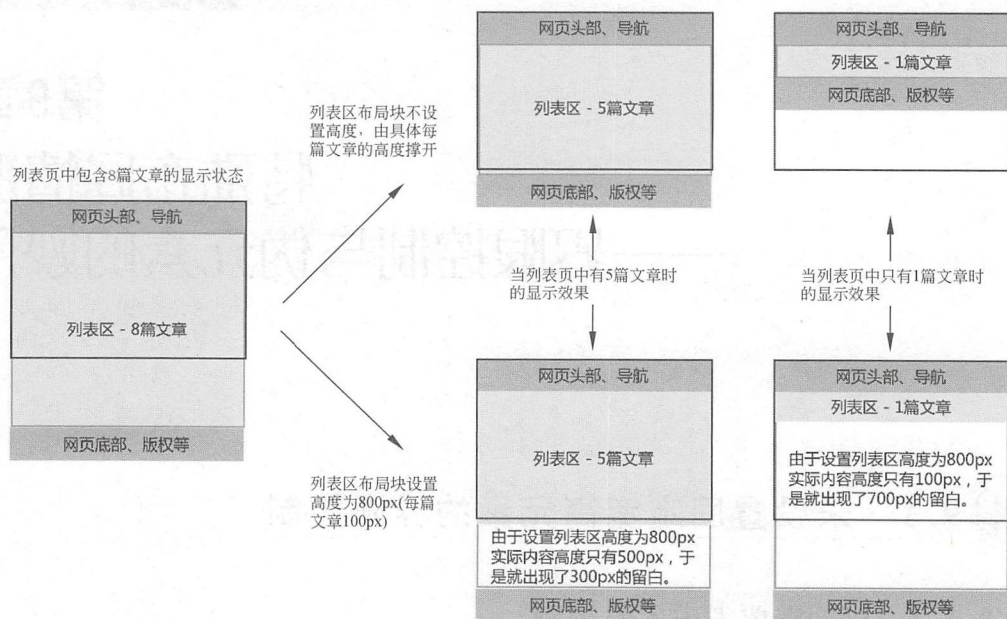


图 9.1 最小高度场景解析

## 代码实例：

```

<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 最小高度</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .head, .foot {
      height: 80px;
      background: # 39f;
    }
    .con - list {
      min - height: 500px;
      border: 2px solid black;
    }
    .con - list li {
      height: 100px;
      background: # ccc;
    }
  </style>
</head>
<body>
  <div class = "head">头部</div>
  <ul class = "con - list">
    <li>具体列表项 01</li>
  </ul>

```



```

        <li>具体列表项 02 </li>
        <!-- <li>具体列表项 03 </li>
        <li>具体列表项 04 </li>
        <li>具体列表项 05 </li>
        <li>具体列表项 06 </li>
        <li>具体列表项 07 </li>
        <li>具体列表项 08 </li> -->
    </ul>
    <div class = "foot">底部</div>
</body>
</html>

```

#### 代码解析:

为 ul(列表项父级元素)设置最小高度为 500px,当元素内容部分的高度小于“最小高度值”时(如内容高度为 200px),该元素按照最小高度值(500px)进行渲染;当元素内容部分的高度大于“最小高度值”时(如内容高度为 700px),该元素按照实际高度值(700px)进行渲染。

注: 打开代码注释,让内容高度发生变化。

### 9.1.3 最小宽度

#### 1. 场景解析

在设计师提供给开发工程师的设计图当中,经常会出现这样的现象:设计图的宽度远大于内容区域的宽度,在内容区外部,并非是完全空白的,而是有一定的背景颜色。那么此时,到底如何解读设计图呢?

如图 9.2 所示,内容区域的宽度只有 990 像素,但是设计师提供的设计图是 1280 像素的宽度,在 990 像素左右各有 145 像素的留白。在两侧的留白区域,有一些背景颜色。

对于这张 PSD 设计图,设计师是想告知开发工程师“内容宽度只有 990 像素,页面中大于 990 像素的部分按照留白区域的样式来展示”。因此,开发工程师应以 990 像素(内容区域最大宽度)开发网页,而不应以 1280 像素作为宽度大小开发网页。

图中攻略区域的整体布局实现代码实例:

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - 最小宽度</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        .arc - box {
            height: 200px;
            background: # f0f0f0;
        }
        .arc - con {
            width: 990px;

```

```
height: 200px;
margin: 0 auto;
border: 1px solid black;

    }
</style>
</head>
<body>
    <div class = "arc - box">
        <div class = "arc - con">此处为内容区域,边框为内容区的边框</div>
    </div>
</body>
</html>
```

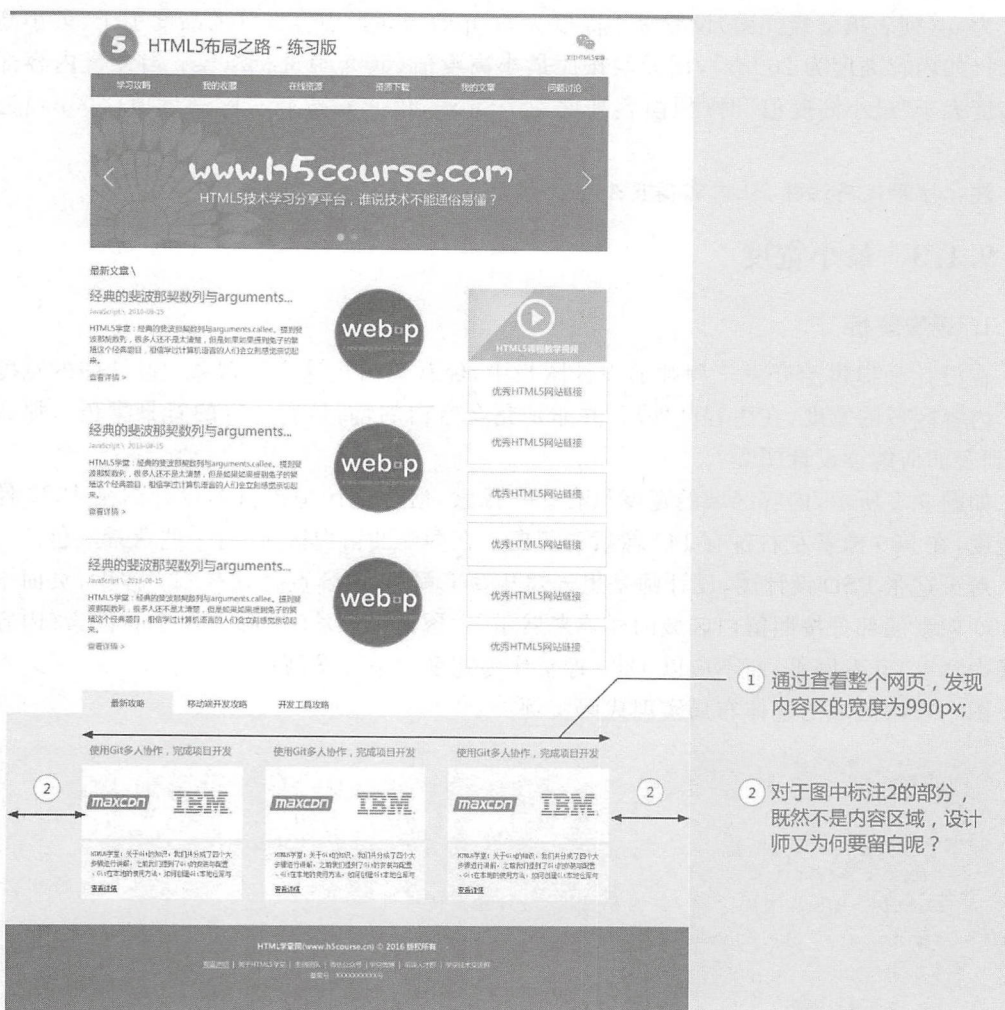


图 9.2 最小宽度场景解析

代码解析：

内容区域宽度大小为 990 像素，而背景颜色应当覆盖整个浏览器宽度，一层标签没有办



法解决需求,需要设置两层标签结构。

外层结构没有设置固定宽度,默认为浏览器的 100%,占满一行。

内层结构设置固定宽度 990 像素,并设置 `margin: 0 auto`,在父级元素当中水平居中。

浏览器窗口大于 990 像素时的显示效果(在该情况下没有任何问题)如图 9.3 所示。

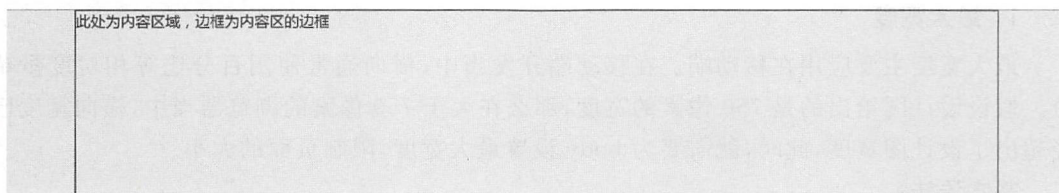


图 9.3 浏览器宽度大于 990px 的显示效果

浏览器窗口小于 990 像素时的显示效果(当拖动滚动条时,右侧并没有背景)如图 9.4 所示。

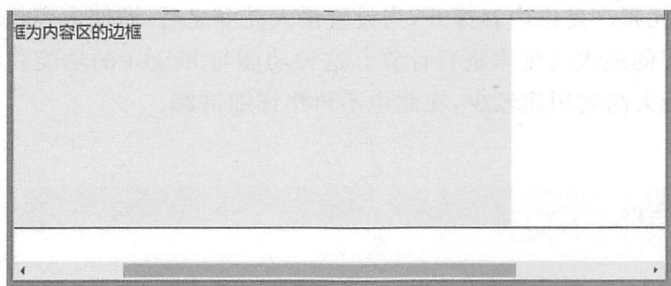


图 9.4 浏览器宽度小于 990px 的显示效果

#### 问题解析:

造成这个问题的原因,就是最外层元素没有固定宽度。

最外层元素的宽度为父级(body)宽度的 100%,也就是等于浏览器可视窗口的宽度。

内容区域为 990 像素,当浏览器宽度大于 990 像素时,最外层元素的宽度也是大于 990 像素,大于内容区域,背景会正常显示;当浏览器宽度小于 990 像素时(假设为 600 像素),最外层元素的宽度只有 600 像素,此时,在内容区域左侧的 600 像素区域具有背景,剩余 390 像素区域则没有背景。

#### 解决办法:

为外部元素设置最小宽度(min-width),最小宽度值通常设置为内容区的宽度大小,此处设置最小宽度为 992 像素(990 像素的宽度以及横向 2 像素的边框)。

### 2. 最小宽度的用法

#### 基本语法:

```
min-width: 992px;  
/* 为元素设置最小宽度为 990 像素 */
```

#### 代码解析:

为外层元素设置最小宽度之后,该元素的宽度在“当前浏览器宽度”与“设置的最小宽

度”当中,取其大者。最终的值均是大于或等于内容区宽度的,自然就不会出现空白背景的问题了。

9.1.4 最大宽度与最大高度

1. 最大宽度

最大宽度主要应用在移动端。在移动端开发当中,横向通常使用百分比等相对度量单位。假设计图给出的是 750 像素的宽度,那么在大于 750 像素的浏览器当中,横向宽度已经超出了设计图宽度,此时,就需要为 body 设置最大宽度,限制页面的大小。

基本语法:

```
max-width: 750px;
```

2. 最大高度

默认情况下,元素高度由内容撑开,当设置最大高度之后,即使内容超出这个高度,浏览器依旧会按照设置的最大高度来进行计算。这种功能与 height 的功能并没有什么区别,因此,在开发当中,最大高度用途较少,在此也不再作详细讲解。

基本语法:

```
max-height: 200px;
```



9.2 伪元素

9.2.1 什么是伪元素

伪元素可以理解成“虚假的元素”,它们虽然会在内容元素的前后插入额外的元素,但并不会在文档中生成,在文档的源代码当中并不能够找到它们。

虽然在结构上是“虚假”的元素,但在表现上和“普通”元素没有什么区别,能够为它们添加任何样式,比如改变文字颜色、添加背景、调整字体大小等。

9.2.2 伪元素的种类

伪元素的种类如表 9.1 所示。

表 9.1 伪元素的种类

伪元素种类	描 述
:first-letter	向文本的第一个字母添加特殊样式
:first-line	向文本的首行添加特殊样式
:before	在元素之前添加内容
:after	在元素之后添加内容



:first-letter 与 :first-line 属于 CSS1.0 中的样式。:before 和 :after 属于 CSS2.1 中的样式,IE7-不支持。

在伪元素当中,:before 和 :after 较为常用,能够辅助实现各类效果和功能。对于 :first-letter 和 :first-line 使用较少,在此不做讲解。

### 9.2.3 伪元素的书写格式

在最初,伪元素的语法是使用“:”(一个冒号)。在 CSS3 当中,通过冒号的数量区分伪元素和伪类,伪元素使用两个冒号(:before、:after),伪类选择器使用一个冒号(:hover、:active)。

无论开发工程师使用一个冒号还是两个冒号,浏览器都将能识别它们。但是 IE8-浏览器只支持一个冒号的格式,因此,如果需要兼容低版本的 IE 浏览器,请使用一个冒号的书写格式。

### 9.2.4 after 与 before 伪元素

使用 :before,将会在选中的元素之前“添加”一个元素;使用 :after,将会在选中的元素之后“添加”一个元素。在伪元素当中添加内容可以使用 content 属性。

需要注意,如果不为伪元素设置 content 属性,伪元素并不会显示。如果希望伪元素能够显示,又不希望伪元素当中存在任何可视文本内容,可以将 content 设置为“\200B”(零宽度的空格)。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - after 与 before 伪元素</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .bookname:before {
      content: "《";
    }
    .bookname:after {
      content: "》";
    }
  </style>
</head>
<body>
  <div class = "bookname">HTML5 布局之路</div>
</body>
</html>
```

代码解析:

在文字的之前和之后分别添加一个书名号,即变成了《HTML5 布局之路》。

默认情况下,after 与 before 伪元素以行元素的特性进行渲染。

浏览器渲染后的结构标签如图 9.5 所示。

```
<div class="box">
  ::before
  "HTML5布局之路"
  ::after
</div>
```

图 9.5 含伪元素的结构在浏览器渲染后的标签效果

### 9.2.5 让伪元素按照块元素特性渲染

如果希望伪元素能够以块元素显示,为其设置 `display: block;` 即可。如果希望它能够设置宽高也能够与当前元素处于同一行,可以使用浮动、定位等。

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 伪元素转换成块元素</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .bookname {
      width: 200px;
      height: 200px;
      margin: 20px;
      border: 2px solid black;
    }
    .bookname:before {
      content: "\200B";
      display: block;
      width: 100px;
      height: 100px;
      background: #39f;
    }
    .bookname:after {
      content: "\200B";
      display: block;
      width: 100px;
      height: 100px;
      background: #ccc;
    }
  </style>
</head>
<body>
  <div class = "bookname"></div>
</body>
</html>
```



### 9.2.6 伪元素实现背景图

代码实例：

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 伪元素实现背景图</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .menu {
      width: 200px;
      height: 191px;
      padding - left: 197px;
      border: 2px solid black;
      font - size: 40px;
      line - height: 191px;
    }
    .menu:before {
      content: "\200B";
      position: absolute;
      top: 0;
      left: 0;
      width: 197px;
      height: 191px;
      background: url('../images/menu.png') 0 0 no - repeat;
    }
  </style>
</head>
<body>
  <div class = "menu">菜单</div>
</body>
</html>
```

显示效果如图 9.6 所示。

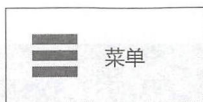


图 9.6 利用伪元素实现背景图的显示效果

### 9.2.7 伪元素的问题区

伪元素与伪类有何区别？

伪类的效果可以通过添加一个实际的类来达到，而伪元素的效果则需要通过添加一个实际的元素才能达到，这也是为什么它们一个称为伪类，一个称为伪元素的原因。

伪元素用于处理结构，伪类用于处理样式。

# 第10章

## 表 格



### 10.1 table 布局的兴衰

#### 10.1.1 表格的发展历史

表格是 HTML 标签中拥有无比辉煌历史的成员,在 2005 年之前,使用 table 表格进行网页布局的方式红遍了大江南北。

一方面,当时网页三剑客之一的 Dreamweaver,其中可视化的操作方式是自动生成 table 布局类型的网页,操作起来比较简单,容易上手。

另一方面,table 布局方式,能够很好地处理各个浏览器的兼容,布局错乱的问题相对较少,表格的单元格还支持垂直方向的居中对齐(vertical-align: middle)。

当然,table 布局也存在着它的问题:层层嵌套导致网页的嵌套深度太深,代码量过大;最重要的一点在于,从语义性(SEO)的角度上来说,table 表示数据表,是通过定义一个表格来呈现具体数据,而非用于布局的工具。

#### 10.1.2 表格的应用

当前的网页开发当中,表格已经走下“布局”的神台,重新回到了“数据呈现”的功能层面当中,通常使用表格来实现网页中的数据表部分。表格在后台管理系统当中的使用极其频繁。

表格应用效果图 1 如图 10.1 所示。

学员信息					
姓名	毕业院校	学历	专业就业	单位	月薪
姓名1	中南大学	本科	生物科学与技术	搜狐	5000元/月
姓名2	河北师范	本科	计算机软件工程	中国药学会科技开发中心	5000元/月
姓名3	天津职业技术	本科	计算机科学与技术	未来付	6000元/月
姓名4	河南工业大学	本科	软件开发	恒德炫艺	6000元/月

图 10.1 表格应用效果图 1



表格应用效果图 2 如图 10.2 所示。

诊断类别	诊断描述	疾病代码	治疗结果	天数	手术	诊断日期
门诊诊断	右髌骨骨折	ABC	ABC		<input checked="" type="checkbox"/>	2016-09-04
入院初诊	右髌骨骨折	ABC	ABC		<input type="checkbox"/>	2016-09-04
主要诊断	右髌骨骨折	ABC	ABC	279	<input type="checkbox"/>	2016-09-04
其他诊断	右髌骨骨折	ABC	ABC	279	<input type="checkbox"/>	2016-09-04
病理诊断	右髌骨骨折	ABC	ABC		<input type="checkbox"/>	2016-09-04
院内感染	右髌骨骨折	ABC	ABC	279	<input type="checkbox"/>	2016-09-04
损伤和中毒的外部原因	右髌骨骨折	ABC	ABC		<input type="checkbox"/>	2016-09-04
根本死亡原因	右髌骨骨折	ABC	ABC		<input type="checkbox"/>	2016-09-04

图 10.2 表格应用效果图 2



## 10.2 table 各类元素以及用法

### 10.2.1 基本标签

在第 2 章当中介绍了表格类的元素,在此再简单罗列一下,如表 10.1 所示。

表 10.1 表格的各种元素

标签	含 义	标签	含 义
table	定义一个表格	col	定义表格的列区域
caption	定义表格标题	colgroup	定义表格的数据列组
thead	定义表格的头部区域	tr	定义一个表格行
tbody	定义表格的主体区域	td	定义一个表格单元格
tfoot	定义表格的脚部区域	th	定义一个表格的表头单元格(列标题)

使用 tr 来表示“一行”,用 th 或 td 表示一个具体的单元格,通常 th 作为 thead 的 tr 当中的子元素,td 作为 tbody 或 tfoot 的 tr 当中的子元素。

col 元素为单标签,col 元素可以嵌套于 colgroup 元素当中,也可以单独使用。使用 colgroup 时必须配合 col 元素,否则无效。在 col 元素当中可以定义 span 属性来指定列分组的数目,默认值为 1。

### 10.2.2 表格的嵌套规则

table 元素下只能直接包含 caption、thead、tbody、tfoot、col、colgroup 元素;thead、tbody、tfoot 下可直接包含 tr,tr 下可包含 th 或 td。通常 th 出现在 thead 的 tr 标签中。

### 10.2.3 行列合并控制

#### 1. rowspan 与 colspan

在表格当中,有些单元格进行了合并处理,即有些单元格占据了多个单元格的空間。如

果想实现这种功能,需要使用到 rowspan 与 colspan 两种属性。

rowspan 表示行合并,colspan 表示列合并,两者是 th 或 td 的属性,属性值为数字,表示的是合并的行数或列数。

在设置合并时,为“需合并单元格”的左上方第一个单元格设置 rowspan 或 colspan 属性,而其他行或列则直接删除掉多余的 td 或 th 元素。

2. 行列合并的数据表实例解析

效果实例如图 10.3 所示。

第1行 第1列	第1行 第2列	第1行 第3列	第1行 第4列	第1行 第5列
第2行 第1列	合并的单元格			第2行 第5列
第3行 第1列				第3行 第5列
第4行 第1列	第4行 第2列	第4行 第3列	第4行 第4列	第4行 第5列

图 10.3 行列合并表格需求

需求分析如图 10.4 所示。

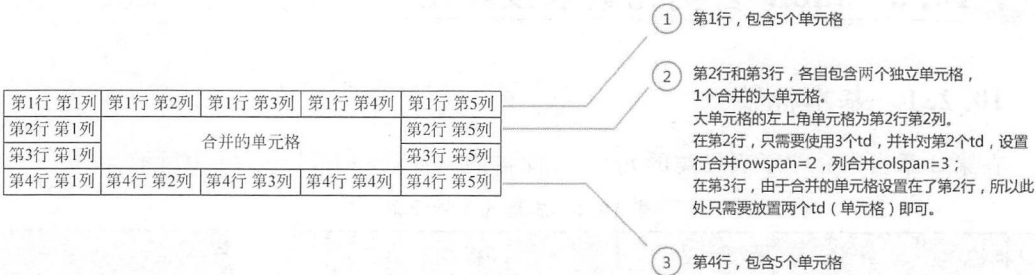


图 10.4 行合并列合并的表格

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 行列合并的表格实例</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .data td {
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <table class = "data">
    <tbody>
      <tr>
        <td>第 1 行 第 1 列</td>
        <td>第 1 行 第 2 列</td>
```



```

        <td>第 1 行 第 3 列</td>
        <td>第 1 行 第 4 列</td>
        <td>第 1 行 第 5 列</td>
    </tr>
    <tr>
        <td>第 2 行 第 1 列</td>
        <td rowspan = 2 colspan = 3>合并的单元格</td>
        <td>第 2 行 第 5 列</td>
    </tr>
    <tr>
        <td>第 3 行 第 1 列</td>
        <td>第 3 行 第 5 列</td>
    </tr>
    <tr>
        <td>第 4 行 第 1 列</td>
        <td>第 4 行 第 2 列</td>
        <td>第 4 行 第 3 列</td>
        <td>第 4 行 第 4 列</td>
        <td>第 4 行 第 5 列</td>
    </tr>
</tbody>
</table>
</body>
</html>

```

### 10.2.4 关于表格元素的问题区

(1) caption 是否是必需的?

caption 不是必需的。

对于一个数据表来说,有可能存在标题,也可能没有标题,因此在没有标题时,是可以省略 caption 元素的,即便存在标题,也可以使用其他元素(如<h2>)实现表格标题。

(2) tbody、thead、tfoot 是否必须书写,且顺序不能更换?

tbody、thead、tfoot 元素非必须书写。

数据表有可能没有表头,也有可能不存在表尾,因此,thead、tfoot 元素都是根据需求而书写的,如果数据表当中没有,则完全可以省略。对于 tbody,也是可以省略的,但是由于使用 table 就是为了呈现数据,因此通常很少出现“没有数据的数据表”这样的需求。

关于 tbody、thead 和 tfoot,在书写时,顺序是可以任意更换的。浏览器会读取标签名,然后将标签“智能地”放置在合适位置。换言之,即便书写顺序为“tfoot-thead-tbody”,显示顺序依旧是“thead-tbody-tfoot”。

代码实例:

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">

```

```
<title>HTML5 布局之路 - tfoot,tbody,thead 的顺序问题</title>
<link rel = "stylesheet" href = "../css/reset.css">
<style>
    .data td, .data th {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <table class = "data">
        <tfoot>
            <tr>
                <td colspan = 3>«HTML5 布局之路» —— HTML5 学堂创始人</td>
            </tr>
        </tfoot>
        <thead>
            <tr>
                <th>HTML5 技术分享平台</th>
                <th>链接地址</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>HTML5 学堂</td>
                <td><a href = "http://www.h5course.com" title = "HTML5 技术原创分享平台">
http://www.h5course.com</a></td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

显示效果如图 10.5 所示。

HTML5技术分享平台	链接地址
HTML5学堂	http://www.h5course.com
《HTML5布局之路》 —— HTML5学堂创始人	

图 10.5 tbody thead tfoot 书写顺序实例效果图

在代码当中,先书写的是 tfoot,之后书写的是 thead,最后书写的是 tbody 的内容。从当前的显示效果当中,不难看出 tfoot、thead、tbody 的顺序并没有对最终效果造成影响。



## 10.3 基本数据表的开发与制作

### 10.3.1 基本数据表的功能需求

某成绩表如图 10.6 所示。



2016年12月 四年级3班期末考试成绩				
姓名	语文	数学	英语	总分
学生姓名1	89	65	77	231
学生姓名2	54	77	87	218
学生姓名3	97	42	45	184
学生姓名4	65	88	78	231
学生姓名5	92	65	98	255
学生姓名6	85	89	67	241
学生姓名7	72	96	66	234
学生姓名8	69	93	60	222
学生姓名9	84	65	45	194
学生姓名10	73	85	39	197
学生姓名11	66	68	96	230
学生姓名12	74	75	84	233
学生姓名13	80	92	56	228
学生姓名14	84	90	79	253
学生姓名15	79	87	88	254
备注：2016年年底期末考试 整体存在偏科现象				

图 10.6 数据表功能需求——成绩表

### 10.3.2 基本数据表的实现思路

#### 1. 确定是否需要使用“table”实现功能需求

table 虽然能够实现所有布局,但是由于 table 的语义为“数据表”,因此,在遇到“多行多列”的“数据表”时才会考虑使用 table 系列元素来实现。

#### 2. 拆分数据表的结构组成

将数据表拆分为“标题部分”、“表头部分”、“表体部分”、“表尾部分”,分别对应于表格元素中的 caption、thead、tbody、tfoot 元素。

#### 3. 构思数据是否有必要划分“列”

在数据表当中,不同列的数据,类型有所不同;在样式操作当中,有时会根据列的种类来设置样式,此时为了便于“列”的样式控制,就会使用到 col 和 colgroup。

#### 4. 确定行数以及每行的单元格个数

确定表格的行列数,并在每行(tr)当中均书写相应列数的单元格(th 或 td)。

#### 5. 查看是否需要合并

查看是否存在跨行跨列的单元格,使用 colspan、rowspan 属性,进行单元格的合并,并在相应的行和列当中删除掉多余的单元格(td 或 th)。

### 10.3.3 基本数据表的需求分析

按照实现思路一步一步进行分析,如图 10.7 所示。

通过对功能需求的分析,结果如下。

采用 table 来实现该功能需求(成绩表);

该成绩表当中包含标题、表头、表体、表尾,需要使用到 caption、thead、tbody、tfoot

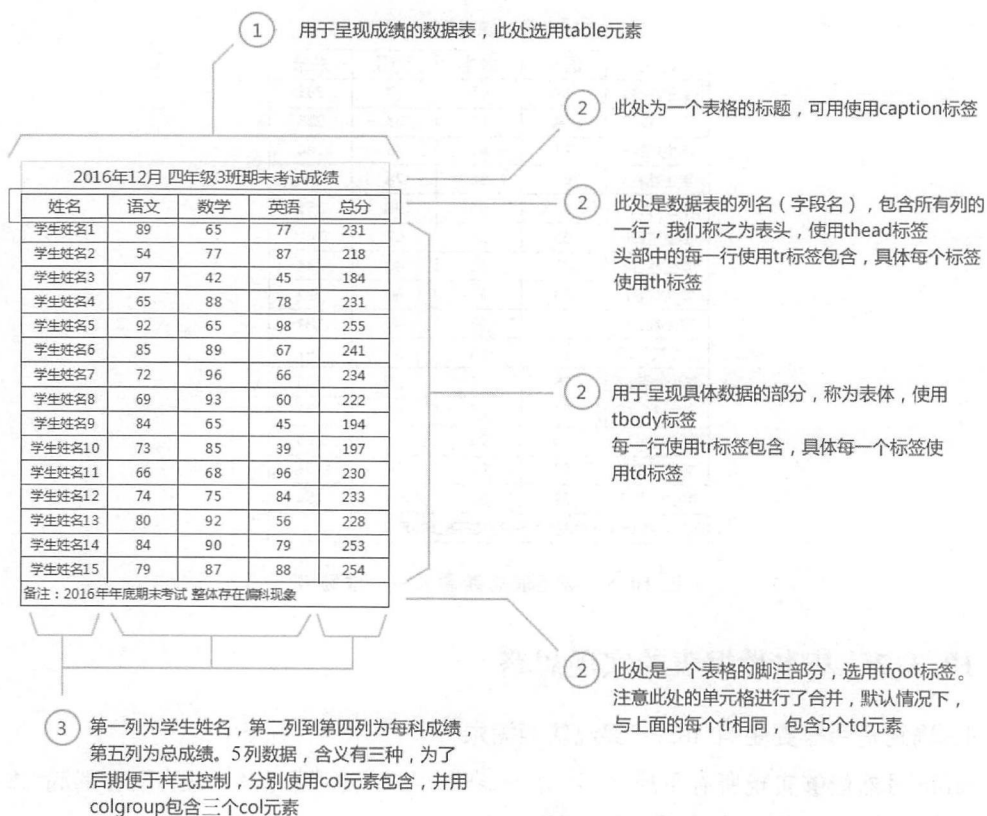


图 10.7 数据表需求分析

标签：

共存在 5 列数据，分为三种类型，需要使用到 col、colgroup 标签；

在表尾部分，存在合并的单元格，进行的是列合并，需要使用到 colspan 标签。

### 10.3.4 基本数据表的实现

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 基本数据表的实现</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .data td, .data th {
      border: 1px solid black;
    }
  </style>
</head>
<body>
```



```

<table class="data">
  <caption>2016 年 12 月 四年级 3 班期末考试成绩</caption>
  <colgroup>
    <col span="1">
    <col span="3">
    <col span="1">
  </colgroup>
  <thead>
    <tr>
      <th>姓名</th>
      <th>语文</th>
      <th>数学</th>
      <th>英语</th>
      <th>总分</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>学生姓名 1</td>
      <td>89</td>
      <td>65</td>
      <td>77</td>
      <td>231</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="5">备注：2016 年年底期末考试 整体存在偏科现象</td>
    </tr>
  </tfoot>
</table>
</body>
</html>

```

备注：出于代码量的考虑，只书写了 tbody 元素当中的第一行数据（“学生姓名 1”的具体数据），其他行的数据则直接省略。



## 10.4 table 元素的属性

### 10.4.1 table 的常见属性

表格的常见属性如表 10.2 所示。

表 10.2 表格的常见属性

属 性	属性功能说明
width	用于定义数据表的宽度
height	用于定义数据表的高度



续表

属 性	属性功能说明
border	用于定义表格边框的宽度,取值为 0 时,表示隐藏边框线
cellpadding	用于定义数据表单元格的补白,即单元格与边框之间的距离
cellspacing	用于定义数据表单元格的边界,即单元格与单元格之间的距离

10.4.2 width 与 height——宽度与高度

1. 表格的默认大小

在默认情况下,表格由内容撑开宽度和高度。各个单元格的总宽度加上单元格边框、空隙的宽度,即为表格的总宽度。

显示效果如图 10.8 所示。

序号	优秀HTML5网站	链接地址
01	HTML5中国	<a href="http://www.html5cn.org">http://www.html5cn.org</a>
02	HTML5学堂	<a href="http://www.h5course.com">http://www.h5course.com</a>
03	W3CSchool	<a href="http://www.w3school.com.cn">http://www.w3school.com.cn</a>

图 10.8 表格的默认大小显示效果

2. 设置 table 的宽度后,单元格宽度的默认状态

当为 table 标签设置宽度之后,各个单元格会按照“单元格内容大小”来“瓜分”table 的所有宽度或高度。

如果当前设置的宽度或高度小于元素的内容的宽度或高度,表格会自动忽略设置的宽度或高度值。

例如,将如上效果,设置 width="600px" height="300px"的属性(设置 width、height 属性时,可以不使用 px,直接使用数字)。显示效果如图 10.9 所示。

序号	优秀HTML5网站	链接地址
01	HTML5中国	<a href="http://www.html5cn.org">http://www.html5cn.org</a>
02	HTML5学堂	<a href="http://www.h5course.com">http://www.h5course.com</a>
03	W3CSchool	<a href="http://www.w3school.com.cn">http://www.w3school.com.cn</a>

图 10.9 设置 table 宽度之后,单元格宽度的默认状态

3. 如何控制每列宽度

从开发的角度来说,为了显示效果的考虑,必然会控制表格当中的每列宽度,此时应当如何操作呢?

方式 1: 为第一行中的每一列的单元格各设置一个样式,赋予不同的宽度。

代码实例:

```
<thead>
  <tr>
    <th class = "col1">序号</th>
    <th class = "col2">优秀 HTML5 网站</th>
```





```
<th class = "col3">链接地址</th>
</tr>
</thead>
```

方式 2: 通过 col 和 colgroup 来进行列分组,并为每个分组各设置一个样式,赋予不同的宽度。

代码实例:

```
<colgroup>
  <col class = "col1">
  <col class = "col2">
  <col class = "col3">
</colgroup>
```

### 10.4.3 border——表格边框设置

此处需要注意的是,这里的 border 是 table 标签的属性,而非 table 的 CSS 样式,虽然 CSS 样式也能够为 table 元素设置边框效果。

代码范例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 表格边框的问题</title>
</head>
<body>
  <table>
    <thead>
      <tr>
        <th>序号</th>
        <th>优秀 HTML5 网站</th>
        <th>链接地址</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>01</td>
        <td>HTML5 中国</td>
        <td><a href = "http://www.html5cn.org" title = "HTML5 论坛"> http://www.html5cn.org </a></td>
      </tr>
      <tr>
        <td>02</td>
        <td>HTML5 学堂</td>
        <td><a href = "http://www.h5course.com" title = "HTML5 技术原创分享平台"> http://www.h5course.com </a></td>
```



```
</tr>
<tr>
  <td>03</td>
  <td>W3CSchool</td>
  <td><a href = "http://www.w3school.com.cn" title = "HTML5 技术知识点'字典'">http://www.w3school.com.cn</a></td>
</tr>
</tbody>
</table>
</body>
</html>
```

当前的显示效果如图 10.10 所示。



图 10.10 表格没有设置边框属性

为 table 标签添加 border 属性,将 border 的属性值设置为 1,即< table border="1">,此时,效果如图 10.11 所示。



图 10.11 表格设置边框属性

#### 10.4.4 cellpadding 与 cellspacing——空白区设置

cellspacing 用于设置单元格与边框之间的距离; cellpadding 用于设置单元格与单元格





之间的距离。

为 table 设置如下属性：border=1 cellspacing=0 cellpadding=0。显示效果如图 10.12 所示。

序号	优秀HTML5网站	链接地址
01	HTML5中国	http://www.html5cn.org
02	HTML5学堂	http://www.h5course.com
03	W3CSchool	http://www.w3school.com.cn

图 10.12 表格 cellspacing 与 cellpadding 空白区设置

## 10.5 表格特有的 CSS 属性

### 10.5.1 合并单元格之间的边框——border-collapse

属性功能：

设置表格的边框是否被合并为一个单一的边框，或者在标准的 HTML 中分开显示。各个浏览器均支持该属性。

基本语法：

```
table {
    border-collapse: collapse;
}
```

代码解析：

让表格两个单元格之间的边框合并为单一的边框。

属性值如表 10.3 所示。

表 10.3 表格 border-collapse 属性的属性值

值	描 述
separate	默认值。边框会被分开。不会忽略 border-spacing 和 empty-cells 属性
collapse	边框会合并为一个单一的边框。会忽略 border-spacing 和 empty-cells 属性
inherit	规定应该从父元素继承 border-collapse 属性的值

显示效果如图 10.13 所示。

序号	优秀HTML5网站	链接地址
01	HTML5中国	http://www.html5cn.org
02	HTML5学堂	http://www.h5course.com
03	W3CSchool	http://www.w3school.com.cn

未设置边框合并 ( border-collapse ) 时的显示效果

序号	优秀HTML5网站	链接地址
01	HTML5中国	http://www.html5cn.org
02	HTML5学堂	http://www.h5course.com
03	W3CSchool	http://www.w3school.com.cn

设置border-collapse: collapse; 时的显示效果

图 10.13 border-collapse 设置的显示效果



### 10.5.2 边框之间的空隙——border-spacing

属性功能：

在“边框分离”模式之下，设置相邻单元格的边框间的距离，该属性兼容 IE8+ 及各个主流浏览器。

基本语法：

```
table{
    border-collapse:separate;
    border-spacing:10px 20px;
}
```

代码解析：

设置两个单元格之间水平间距为 10 像素，垂直间距为 20 像素。

属性值如表 10.4 所示。

表 10.4 表格 border-spacing 属性的属性值

值	描 述
length length	规定相邻单元的边框之间的距离。使用 px、cm 等单位。不允许使用负值。 如果定义一个 length 参数，那么定义的是水平和垂直间距。 如果定义两个 length 参数，那么第一个设置水平间距，而第二个设置垂直间距
inherit	规定应该从父元素继承 border-spacing 属性的值

显示效果(border-spacing: 10px 20px;)如图 10.14 所示。

序号	优秀HTML5网站	链接地址
01	HTML5中国	<a href="http://www.html5cn.org">http://www.html5cn.org</a>
02	HTML5学堂	<a href="http://www.h5course.com">http://www.h5course.com</a>
03	W3CSchool	<a href="http://www.w3school.com.cn">http://www.w3school.com.cn</a>

图 10.14 border-spacing 设置的显示效果

备注：该属性与 cell-spacing 类似，所不同的是，该属性为 CSS 属性，且能够分别为两个方向设置间距值。而 cell-spacing 属于标签属性，且规定水平与垂直的间距值相同。

### 10.5.3 空白单元格——empty-cells

属性功能：

在“边框分离”模式之下，设置是否显示表格中的空单元格，该属性兼容 IE8+ 及各个主流浏览器。

基本语法：

```
table {
    border-collapse: separate;
```





```
empty-cells:hide;
}
```

#### 代码解析：

不绘制空单元格周围的边框。

属性值如表 10.5 所示。

表 10.5 表格 empty-cells 属性的属性值

值	描 述
hide	不在空单元格周围绘制边框
show	在空单元格周围绘制边框。默认
inherit	规定应该从父元素继承 empty-cells 属性的值

显示效果如图 10.15 所示。

序号	优秀HTML5网站	链接地址
01	HTML5中国	<a href="http://www.html5cn.org">http://www.html5cn.org</a>
02	HTML5学堂	<a href="http://www.h5course.com">http://www.h5course.com</a>
03	W3CSchool	

未设置empty-cells: hide;的显示效果

序号	优秀HTML5网站	链接地址
01	HTML5中国	<a href="http://www.html5cn.org">http://www.html5cn.org</a>
02	HTML5学堂	<a href="http://www.h5course.com">http://www.h5course.com</a>
03	W3CSchool	

设置empty-cells: hide;的显示效果

图 10.15 empty-cells 设置的显示效果



## 10.6 表格属性与样式选用原则

在表格当中,样式既可以通过表格类属性进行设置,又可以通过 CSS 样式进行设置,那么在日常开发时,应当如何选用和设置呢?

请在选用时遵循如下原则。

(1) 以 CSS 为主;

(2) 不为 table 标签设置 width、height、border 属性,使用 CSS 来实现表格宽高、边框样式的设置;

(3) 在使用表格元素之前先清除默认的样式,出于兼容的考虑(兼容 IE7 时),需要同时设置 cellspacing、cellpadding 属性以及 CSS 中的 border-spacing 或 border-collapse 属性。(在 CSS 重置文件当中,有针对表格的样式重置代码)。



# 第11章

## 表 单



### 11.1 认识表单

#### 11.1.1 表单的作用——实现对话

客户端(每个用户的计算机)在浏览网页时,都会向服务器(后台)端索要数据,然后将得到的数据呈现在浏览器当中。除了索要数据之外,有时客户端也希望能够向服务器端发送一些数据。

**备注:**在后来 JavaScript 广泛使用之后,AJAX 技术也能够实现客户端与服务器端的对话。但是在最初的网页当中,表单是唯一能够实现客户端和服务器端“对话”的工具。

#### 11.1.2 向服务端发送数据的场景示例

在此举出几个“从客户端向服务器端发送数据”的场景,以便于理解表单的作用和应用。当然,在实际网站中并不仅局限于这几种场景。

##### 1. 场景 1: 搜索信息

在百度搜索当中,用户需要在搜索框当中输入一些内容,之后单击“搜索”按钮或按下回车键,就可以看到这个“内容”相关的搜索结果。在这个过程中,就涉及数据的发送与获取,具体的流程如图 11.1 所示。

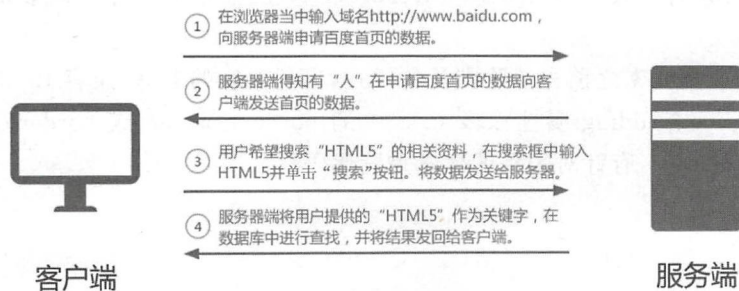


图 11.1 客户端向服务器端索要数据

##### 2. 场景 2: 注册登录

不少网站都提供了注册登录的功能。对于浏览者来说,只是填写了一下具体信息,单击





了“注册”或“登录”的按钮，为何就被网站“记住”了呢？

在这个过程中，注册就是将自己填写的具体信息（用户名、密码等）发送给了服务器。服务器接收到数据之后，如果数据没有和别人的重复，则保存在了服务器的数据库当中，并告知用户“注册成功”；如果数据和别人重复，不能够注册，服务器的数据库并不会存储注册信息，会告知用户“注册失败”。

登录时，浏览者填写用户和邮箱，之后发送给服务器，服务器将这个信息与数据库中的数据进行比较，如果存在于数据库当中，则允许浏览者登录；如果不存在于数据库当中则告知浏览者登录失败，如图 11.2 所示。

图 11.2 登录界面中的表单效果

### 3. 场景 3: 购物车

在各大网站当中的购物车，其实也大量应用了表单，在图 11.3 当中，能够看到的表单部分，包括“促销优惠”的下拉单、“数量”的操作、每件商品前面的复选框（对勾部分）、右下角“去结算”的按钮。

图 11.3 购物车中的表单效果

这些都是要告知服务器端，浏览者要购买哪件商品、购买几件、是否有什么特殊活动等。如果没有这些表单元素的帮助，服务器端是不可能知道每个用户具体想购买的内容的。

当然，在这个页面当中还存在大量的“隐藏域（表单元素的一种）”，也是为了将尽可能完整的信息传递给服务器端。

#### 11.1.3 表单的基本结构

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 表单基本结构</title>
  <link rel = "stylesheet" href = "../css/reset.css">
</head>
```



```
<body>
  <!-- form 为表单元素包含框 -->
  <form action="adduser.php" method="post">
    <!-- 第一个 input 为文本输入框 -->
    <input type="text" name="user" id="user">
    <!-- 第二个 input 为密码框 -->
    <input type="password" name="pass" id="pass">
    <!-- 第三个 input 是提交按钮 -->
    <input type="submit" name="btn" id="btn" value="提交数据">
  </form>
</body>
</html>
```

#### 代码解析：

form 表示表单元素，form 中的第一个 input 是一个文本框，第二个 input 是一个密码框，第三个 input 是“提交”按钮。用户可以在第一个和第二个 input 当中输入内容，之后通过单击“提交”按钮，将数据提交给后台。

一个可以与后台进行数据交互的表单，至少要包含“form”、输入框（不仅局限于 input）、“提交”按钮。

简要了解基本结构之后，在下面几节中来详细介绍每个表单元素和它们的属性。

### 11.1.4 各类表单元素通用属性

id：该属性有两个作用，一方面是便于前端 JavaScript 语言对其进行控制，另一方面是与 label 标签中的 for 属性挂钩（label 在稍后会进行讲解）。

name：连接前端和后台的“桥梁”。后台服务器通过该属性的属性值来读取其中的具体数据或控制该元素。除了按钮之外，其他表单元素都应当设置 name 属性，否则该元素的数据后台无法获取到。

value：用于定义输入框所包含的默认字符串。

备注：对于不同种类表单元素的特有属性，会伴随相应元素来讲解。



## 11.2 表单常用元素

### 11.2.1 form

#### 1. 标签含义

form 元素表示表单，是一个包含框，负责数据的处理，包括各个表单域的数据采集、打包、发送到指定服务器目标文件等功能。

form 元素和 table 元素类似，是一类元素的包含框，即所有的表单元素应当放置在 form 元素当中。只有放置于 form 元素中的表单元素，才能够正常工作，元素中的数据才能够被提交到服务器端。

由于单纯书写 form 时，并没有显示效果，因此 form 元素的代码实例会在 input 中进行



展示。

## 2. form 元素特有属性

form 元素存在如下几种特有属性。

### 1) action

action 属性用于设置将数据提交给“谁”，即表单提交数据的目标文件。进行表单提交的目的通常是为了让后台对用户提交的数据进行处理，因此，这个目标文件应当是包含服务器端脚本的处理文件，通过这个文件接收并处理数据。

### 2) method

method 属性用于设置表单数据的提交方法。共包括两种，分别是 get 和 post。

get 方法：更常用，更方便；性能好；明文发送数据（数据通过 URL 地址传递），没有 post 安全；由于 URL 有一定长度限制，因此传递数据的长度有限。

post 方法：使用相对较少；性能只有 get 的 1/3 左右；由于密文发送数据（不通过 URL 地址），因此比 get 稍微安全一点儿；没有传输数据大小限制。

在实际开发当中，并不用发愁到底应该采用哪种方法，因为决定采用具体方法的是后台，开发工程师只需要遵从于后台的“指示”即可，如果后台告知前端需要用 get 发送数据，那么就使用 get，如果告知需要使用 post 方法，那么就使用 post，如果告知前端两种方法均可使用，那么优先选择 get 方法。

### 3) enctype

enctype 属性用于设置表单中用户输入的数据发送到服务端时采用的编码类型。简言之，就是设置打包表单数据的方法。

该属性包括三种属性值，分别是 application/x-www-form-urlencoded；multipart/form-data；text/plain。

application/x-www-form-urlencoded；将表单中的数据编码成“名值对”的形式发送给服务器。名值对是 JavaScript 中的相关知识，可以看如下代码实例的效果，在此不作介绍。

multipart/form-data；将表单中的数据编码为一条消息，每个表单域对应于消息中的一个部分，然后发送给服务器。

text/plain；将表单数据以纯文本的形式进行编码，不建议使用。

代码实例：

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - form 表单数据提交方法</title>
  <link rel = "stylesheet" href = "../css/reset.css">
</head>
<body>
  <form action = "adduser.php" method = "post" name = "data" enctype = "application/x-www-form-urlencoded;">
    <p><input type = "text" name = "user" id = "user" value = "独行冰海"></p>
    <p><input type = "password" name = "pass" id = "pass" value = "123456789"></p>
    <p><input type = "submit" name = "btn" id = "btn" value = "提交数据"></p>
```

```

    </form>
  </body>
</html>

```

此时数据打包为如下的“名值对”方式：

```
"user = "刘国利", pass = "123456789";
```

如果将上面的 enctype 修改为 multipart/form-data;数据打包为：

```

data = {
  user = {"刘国利"},
  pass = {"123456789"}
}

```

## 11.2.2 input

### 1. 多种形式的输入框

input 是一个“神奇”的元素，它并不仅仅拥有一种表现形式，可以定义多种形式的输入框，用来让用户“输入”数据。输入框的形式由该元素的 type(类型)属性来设置。

代码实例：

```

<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - input - 各类标签</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .form - data p {
      margin: 0 0 8px;
    }
  </style>
</head>
<body>
  <form class = "form - data" action = "adduser.php" method = "post" name = "data" enctype =
"application/x - www - form - urlencoded;">
    <p><input type = "text" name = "user" id = "user" value = "HTML5 布局之路"></p>
    <p><input type = "password" name = "pass" id = "pass" value = "123456789"></p>
    <p><input type = "hidden" name = "" id = "" value = "不显示在页面当中的隐藏域"></p>
    <p>
      <span>单选框 1:</span><input type = "radio" name = "" id = "">
      <span>单选框 2:</span><input type = "radio" name = "" id = "">
    </p>
    <p>
      <span>复选框 1:</span><input type = "checkbox" name = "" id = "">
      <span>复选框 2:</span><input type = "checkbox" name = "" id = "">
    </p>
  </form>

```



```

</p>
<p>文件上传: <input type="file" name="" id=""></p>
<p>图像按钮: <input type="image" name="" id="" alt="图像加载错误时显示该值"
value="无 alt 时显示该值" src="../../images/HTML5.jpg"></p>
<p><input type="button" name="" id="" value="普通按钮"></p>
<p><input type="reset" name="" id="" value="重置按钮"></p>
<p><input type="submit" name="btn" id="btn" value="提交按钮"></p>
</form>
</body>
</html>

```

显示效果如图 11.4 所示。

## 2. input 通用属性

type: type 用于定义该 input 的具体类型。输入框主要包括文本域、密码框、单选框、复选框、隐藏域、文件域、图像域以及三类按钮(普通按钮、重置按钮和提交按钮)。

## 3. 文本域

基本语法:

```
<input type="text" name="user" id="user" value="HTML5
布局之路">
```

代码解析:

type="text" 表示文本域,即普通的文本输入框。用户可以在该元素当中输入内容,这些内容会按照输入的原始效果显示。

相关属性:

value 属性,用于定义文本框当中包含的默认字符串,不书写则没有。

size 属性,用于定义文本框的字符数,即文本框的宽度,但是在开发中并不会使用该属性,而是通过 CSS 来设置文本域的宽度。

maxlength 属性:用于定义文本框当中能够“接收”的最大的字符数。

## 4. 隐藏域

基本语法:

```
<input type="hidden" id="" value="要传递的值">
```

代码解析:

type="hidden" 表示隐藏域,其中的内容并不会显示在网页当中。

虽然它并不会显示在网页当中(用户无法看到),但是在应用中却有着非常重要的作用,毕竟在网页当中有很多数据并不需要用户输入或操作,但是依旧需要作为参数传递给服务器端。

例如,购物车的页面当中,只给用户提供了“数量”、“是否选中”、“促销下拉菜单”三种表



图 11.4 各个表单元素

单元素进行交互。对于每个商品来说,“具体商品的名称、编码、价格、颜色/类型”等内容都是需要传递给后台的,但是并不需要让用户来操作这些方面的内容,因此使用隐藏域,既保证了数据传递,又不会受到用户不必要操作的影响。

相关属性:

不存在 size、maxlength 属性,可以使用 value 属性。value 属性是隐藏域的重要组成部分,该属性内部的具体属性值不能够被用户修改,因此,开发工程师可以利用该属性向服务器端传递各种固定参数。

购物车中的代码显示效果如图 11.5 所示。

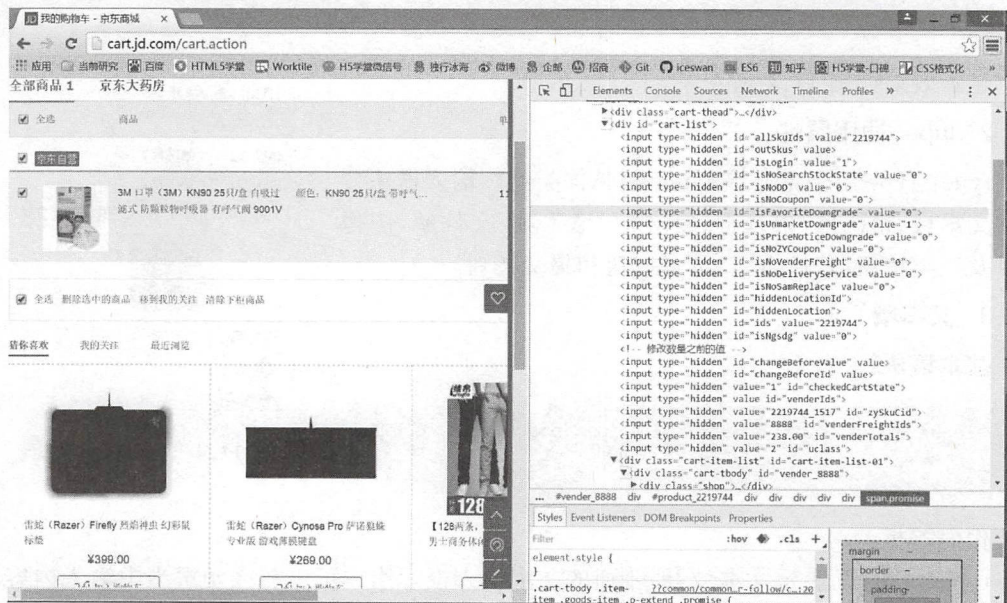


图 11.5 购物车中的隐藏域

## 5. 密码框

### 基本语法:

```
<input type = "password" name = "pass" id = "pass" value = "HTML5">
```

代码解析:

`type="password"` 表示密码框,当用户在该元素当中输入内容时,这些内容并不会直接显示出来,而是以星号或圆点的方式替代文本(不同浏览器显示方式不同)。

相关属性:

密码框与文本域所包含的属性完全相同,在此不再重复讲解。

## 6. 单选按钮

### 基本语法:

```
<input type = "radio" name = "gender" id = "gender" value = "男">
```



### 代码解析:

`type="radio"`表示单选按钮(单选框),单选按钮在表现上是一个圆形的选择框。多个单选按钮可组合为一个“按钮组”。只有被选中的单选按钮,数据才会传递给服务器端。

使用单选按钮的主要目的在于提高提交数据的准确性,简化用户输入数据的步骤,让操作变得更简单快捷。

举个例子:人的性别包括“男”和“女”,此时就需要使用两个单选按钮分别表示“男”和“女”,并将这两个单选按钮组合成一个“按钮组”,组合的方式是将两个 `input` 元素的 `name` 属性设置为同一个属性值。

### 相关属性:

`name` 属性至关重要。对于单选按钮组来说,只能够有一个单选按钮被选中,此时就需要把这个按钮组中所有按钮的 `name` 属性设置为同一个属性值。否则,不能够实现“单选”的需求。

`checked` 属性,单选框、复选框特有属性,能够设置在默认状态下,单选框是否被勾选。(详见本章后面的“表单元素的特殊状态属性”)。

## 7. 复选框

### 基本语法:

```
<input type="checkbox" name="hobby" id="hobby" value="旅游">
```

### 代码解析:

`type="checkbox"`表示复选框(或多选框),复选框在表现上是一个方形的选择框。多个复选框组合在一起可以多选。只有被选中的复选框,数据才会传递给服务器端。

使用复选框的主要目的在于提高提交数据的准确性,简化用户输入数据的步骤,让操作变得更简单快捷。

### 相关属性:

`checked` 属性,单选框、复选框特有属性,能够设置在默认状态下,复选框是否被勾选。(详见后面章节中的“表单元素的特殊状态属性”)。

## 8. 文件域

### 基本语法:

```
<input type="file" name="" id="">
```

### 代码解析:

`type="file"`表示文件域,它包含文本框和“浏览”按钮。用户可以将计算机本地的各类文件传递给服务器。网页浏览者能够通过文件域中的“浏览”按钮,打开“选择文件”的对话框,之后,选择需要上传到服务器端的文件,在选择之后,文件的路径会被自动输入到文本域当中,当提交数据时,浏览器会自动根据这个路径把文件上传到服务器端。

### 注意:

当表单当中存在文件域时,必须将 `form` 元素的 `enctype` 属性设置为“`multipart/form-data`”,`method` 属性设置为“`post`”,否则提交操作会失败。

## 9. 图像域

### 基本语法:

```
<input type="image" name="" id="" alt="图像加载错误时显示该值" value="无 alt 时显示该值" src="../../images/HTML5.jpg">
```

### 代码解析:

type="image"表示图像域,用于定义图像按钮,图像按钮是普通按钮的自定义形式,开发工程师可以通过指定一个图标来定制按钮的样式。

### 相关属性:

图像域包含 alt、src 等属性。其中,alt 属性用于设置替代性文本,src 用于设置图像的路径。

当同时设置 alt 和 value 值,图片加载错误时,显示的是 alt 的内容;如果没有设置 alt 仅设置了 value 属性,在图片加载错误时,显示 value 的内容(谷歌等主流浏览器下)。

### 注意:

在实际开发当中,图像域使用的很少,这是由于在实际开发当中,能够使用 CSS 配合表单元素、其他标签,实现“各种样式”的按钮。

## 10. 按钮

常见的表单按钮包括三种,分别是提交按钮、重置按钮、普通按钮。(图像域也属于按钮的一种,但是和这三种按钮的表现并不相同,所以单独针对图像域进行了讲解)。

### 基本语法:

```
<input type="button" name="" id="" value="普通按钮">
<input type="reset" name="" id="" value="重置按钮">
<input type="submit" name="btn" id="btn" value="提交按钮">
```

### 代码解析:

type="button"表示普通按钮,普通按钮没有动作(即没有特殊功能),需要用户通过 JavaScript 语言为其定义单击后具体的操作。

type="reset"表示重置按钮,当浏览者单击重置按钮之后,所有表单中用户输入的数据都会被清空,恢复到默认状态。

type="submit"表示提交按钮,当浏览者单击提交按钮之后,会将当前表单中的所有数据发送给服务器端。

### 11.2.3 label

#### 标签含义:

<label for="user">用户名</label>表示:定义“id 名为 user 的表单域”提示信息为“用户名”。label 元素最重要的一个作用,是通过 for 属性与表单域的 id 属性,将两者关联起来,此时当用户单击标签文本时,系统会自动把对应的表单域聚焦。

这种功能对于浏览者来说,会让网页的访问变得更加“方便快捷”,用户体验会更好。



代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - label</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .form - data p {
      margin: 8px 0;
    }
  </style>
</head>
<body>
  <form class = "form - data" action = "" method = "" name = "data" enctype = "">
    <p><label for = "user">用户名: </label><input type = "text" name = "username" id =
"user" value = "HTML5 布局之路"></p>
  </form>
</body>
</html>
```

287

代码解析：

为 label 标签设置 for 属性,for 的属性值为 user; 为文本域设置 id 属性,id 的属性值为 user。

这种情况下,当用户单击 label 标签中的“用户名”时,文本域会处于聚焦状态。

单击 label 标签中的文本后,网页的显示效果如图 11.6 所示。

用户名:

图 11.6 单击 label 标签文本后,网页的显示效果

## 11.2.4 select、option 与 optgroup

### 1. 下拉菜单与选项

标签含义：

select 元素用于定义下拉菜单或列表框,默认情况下显示为下拉菜单,如果为 select 设置 size 属性,则会以列表框的形式显示。

select 元素必须与 option 元素配合使用,通过 option 元素定义每个选项的信息,包含显示的值和要传递的值。

基本语法：

```
<select name = "" id = "">
  <option value = "HTML5 布局之路">HTML5 布局之路</option>
</select>
```

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - select、option</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .form - data p {
      margin: 8px 0;
    }
  </style>
</head>
<body>
  <form class = "form - data" action = "" method = "" name = "" enctype = "">
    <p>
      <span>选择职位: </span>
      <select name = "" id = "">
        <option value = "HTML5 开发工程师">HTML5 开发工程师</option>
        <option value = "Web 前端开发工程师">Web 前端开发工程师</option>
        <option value = "JS 开发工程师">JS 开发工程师</option>
      </select>
    </p>
    <p>
      <span>选择职位: </span>
      <select name = "" id = "" size = "3" multiple = "multiple">
        <option value = "HTML5 开发工程师">HTML5 开发工程师</option>
        <option value = "Web 前端开发工程师">Web 前端开发工程师</option>
        <option value = "JS 开发工程师">JS 开发工程师</option>
      </select>
    </p>
  </form>
</body>
</html>
```

显示效果如图 11.7 所示。

相关属性：

size 属性,设置 size 属性之后,select 元素会以列表框的形式来展示。如果将 size 值设置为 1,则展示状态与下拉菜单一样。如果 size 值大于或等于 select 元素所包含的 option 元素,则列表框并不会出现滚动条。

multiple 属性,用于设置列表框是否允许多选。设置 multiple 属性之后,select 元素也会以列表框的形式来展示,该属性值即为 multiple,设置之后,可以按下 Ctrl 键,使用鼠标选择多个列表选项。特有属性: selected(详见后面章节中的“表单元素的特殊状态属性”)。

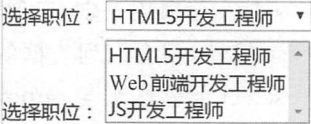


图 11.7 下拉菜单



## 2. select 选项分组

### 标签含义：

optgroup 元素用于在下拉菜单或列表框当中定义分组选项,根据需求情况来使用,在当前开发当中使用相对较少。

需要对选项分组时,将相应组的 option 放置于同一个 optgroup 当中。可以将选项分为多个组。

### 基本语法：

```
<select name = "" id = "">
  <optgroup label = "前端书籍">
    <option value = "HTML5 布局之路">HTML5 布局之路</option>
  </optgroup>
</select>
```

### 代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - select,option</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .form - data p {
      margin: 8px 0;
    }
  </style>
</head>
<body>
  <form class = "form - data" action = "" method = "" name = "" enctype = "">
    <p>
      <span>select 选项分组: </span>
      <select name = "" id = "">
        <optgroup label = "职位名">
          <option value = "HTML5 开发工程师">HTML5 开发工程师</option>
          <option value = "Web 前端开发工程师">Web 前端开发工程师</option>
          <option value = "JS 开发工程师">JS 开发工程师</option>
        </optgroup>
        <optgroup label = "博客名">
          <option value = "独行冰海">独行冰海</option>
          <option value = "梦幻雪冰">梦幻雪冰</option>
        </optgroup>
      </select>
    </p>
  </form>
</body>
</html>
```

显示效果如图 11.8 所示。

相关属性：

label 属性, 使用于 optgroup 标签, 用于定义当前分组的分类名称(该分类名称不能够被选中)。

### 11.2.5 textarea

标签含义：

textarea 标签用于设置多行文本域。如果希望用户输入大量的数据, 可以选择该标签。

基本语法：

```
<textarea name = "" id = "">textarea 具体内容</textarea>
```

展示效果如图 11.9 所示。

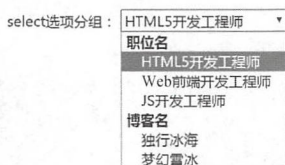


图 11.8 下拉菜单选项分组

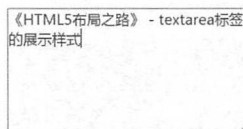


图 11.9 多行文本域

相关属性：

textarea 标签包含多种属性, 但是在实际开发当中使用较少, 在这里仅做了解即可。

cols 属性与 rows 属性: 用于设置文本区域的字符宽度和行数, 即以字符为单位, 定义文本域的宽度和高度(例如将 cols 设置为 30, 就表示 30 个字符大小的宽度)。由于 CSS 也能够控制标签的宽高, 而且操作方式更灵活, 所以并不建议在开发中使用 cols 和 rows 属性。

wrap 属性: 用于定义用户输入内容大于文本区域宽度时的显示方式, 包括三种状态。

(1) 默认状态(virtual): 文本会自动换行, 且在数据被提交时, 自动换行的位置并不会包含额外的换行符信息。

(2) off: 关。禁止文本自动换行, 当输入的文本超出文本域右边界时, 文本会自动滚动, 使用 Enter 键可以实现手动换行。

(3) physical: 实体。允许文本自动换行, 且在数据被提交时, 自动换行位置的换行符也会被一起提交给服务器端。

### 11.2.6 button

标签含义：

<button>标签用于定义一个按钮。

基本语法：

```
<button type = "button">单击按钮</button>
```



相关属性:

button 可以为 type 属性设置“button”、“reset”、“submit”等属性值,来控制按钮的类型。

如果不为 button 按钮定义 type 属性的属性值,在 IE7-浏览器当中默认为“button”,在其他浏览器中(包括 W3C 规范)默认为“submit”。

### 11.2.7 fieldset 与 legend 元素

#### 1. fieldset 与 legend 的基本含义

fieldset 表示对表单进行分组。

legend 表示 fieldset 标签的标题,包含于 fieldset 元素当中(legend 与 fieldset 的关系可以简单理解成 dt 和 dl 的关系)。

#### 2. fieldset 与 legend 的作用

首先,编写网页的时候要考虑到设备屏幕很小的情况。如果试图把一个大的 form 在一个屏幕里面显示,就要指定那些标签保持在一个屏幕中,那么 fieldset 标签即可以将这些标签捆绑在一个屏幕上。

另外,fieldset 标签将表单内容的一部分打包,生成一组相关表单的字段,也就是所谓的分组。例如,可以将注册信息分成“基本信息(一般为必填)”和“详细信息(一般为可选)”两组,如何更好地来实现呢?此时,可以考虑在表单 form 中加入 fieldset(对表单进行分组,一个表单可以有多个 fieldset)和 legend(说明每组的内容描述)。

#### 3. fieldset 与 legend 的默认样式

fieldset 默认存在边框,legend 默认显示在左上角。在某些页面中,不希望让 fieldset 和 legend 的默认样式或默认布局影响设计方案中的样式或布局。解决方法是使用 CSS 的“重置”,将不同浏览器的默认显示效果清除掉,使用其他的元素替代 legend 元素来实现想要的效果。

### 11.2.8 表单元素的问题区

input(按钮类型)与 button 的区别如下。

标签表现方面:input 为单标签,button 为双标签。

标签功能方面:button 标签要比 input 的按钮功能更加强大,在 button 标签当中能够包含其他标签(还可以是图像标签),button 标签的 value 值并不是其属性,而是标签的文本内容。

浏览器提交数据方面:如果在 HTML 表单中使用 button 元素,在前端向后台提交数据时,不同的浏览器会提交不同的值。IE7-浏览器将提交< button>与</button>之间的文本,而其他浏览器将提交 value 属性中的内容。

基于如上的原因,建议在实际开发当中,使用 input 元素来创建表单中的按钮。



### 11.3 表单嵌套规则

最初 HTML 标签刚刚出现时,要求 form 元素不能够直接包含 input 等表单元素,而必须先包含一个块状元素(如 fieldset)。在当今的开发当中,表单的嵌套越来越灵活,因此可以根据具体情况来调整嵌套。不过,还是建议在 form 元素当中嵌套 div、fieldset 等块元素,更便于表单元素的布局。



### 11.4 表单元素的特殊状态属性

#### 1. disabled 与 readonly

如果为表单元素设置 disabled,则表示元素不可被操作,如果为表单元素设置 readonly,则表示元素处于只读状态,如表 11.1 所示。

表 11.1 表单元素的 disabled 与 readonly 属性对比

具体属性	disabled	readonly
文本编辑情况	不能够被编辑	不能够被编辑
属性针对的元素类型	所有的表单元素都有效,包括 select, radio, checkbox, button 等	input ( text、 password ) 和 textarea 有效
数据传递	在将表单提交数据时,该元素的值不会被传递出去	在表单提交数据时,该元素的值会被传递出去
焦点获取	表单输入项不能获取焦点用户的所有操作(鼠标单击和键盘输入等)对该输入项都无效	表单元素依旧可以获取焦点

只要为元素设置了 disabled 或 readonly 属性,那么其值即为 true。如下前三种方法等价,后三种方法等价。

基本语法:

```
disabled = " disabled"
disabled = "true"
disabled
readonly = "readonly"
readonly = "true"
readonly
```

代码实例:

```
<form action = "">
  <div>
    <input type = "text" readonly = "readonly" value = "HTML5 布局之路" name = "" id = "">
    <input type = "submit" disabled value = "单击按钮">
  </div>
</form>
```



## 2. selected

selected 属性是下拉菜单的特有属性,用于设置当前 option 元素是否处于选中状态。只要为 option 元素设置了 selected 属性,那么其值即为 true。如下几种设置方法等价。

**基本语法:**

```
selected = "selected"
selected = "true"
selected
```

**代码实例:**

```
<form action = "">
  <div>
    <select name = "" id = "">
      <option value = "HTML5 学堂">HTML5 学堂</option>
      <option value = "独行冰海">独行冰海</option>
      <option selected value = "梦幻雪冰">梦幻雪冰</option>
    </select>
    <input type = "submit" value = "单击按钮">
  </div>
</form>
```

在页面初始状态中,被设置 selected 的第三个 option,就会作为默认选中项,显示在页面当中。

## 3. checked

checked 属性是单选框和复选框的特有属性,用于设置单选框或复选框是否处于选中状态。只要为元素设置了 checked 属性,那么其值即为 true。如下几种设置方法等价,都可以设置这个当前的 checkbox 或 radio 为选中状态。

**基本语法:**

```
checked = "checked"
checked = "true"
checked
```

**代码实例:**

```
<form action = "">
  <div>
    <label for = "">男</label><input type = "radio" checked = "true" name = "gender"
id = "">
    <label for = "">女</label><input type = "radio" name = "gender" id = "">
  </div>
  <div>
    <label for = "">男</label><input type = "radio" checked = "checked" name = "gender2"
id = "">
```

```
<label for = "">女</label><input type = "radio" name = "gender2" id = "">
</div>
<div>
  <label for = "">男</label><input type = "radio" name = "gender3" id = "">
  <label for = "">女</label><input type = "radio" checked name = "gender3" id = "">
</div>
</form>
```



## 11.5 属性选择器

### 11.5.1 属性选择器的应用场景

属性选择器较多地应用于“处理不同类型 input 的样式”当中，虽然在其他标签当中也能够使用到属性选择器，但是相对较少，IE7+以及各个主流浏览器对该种选择器均有着良好的支持。

### 11.5.2 基本的属性选择器

基本属性选择器如表 11.2 所示。

表 11.2 基本属性选择器

选择器语法	选择器含义
E[att]	包含属性 att 的 E 元素
E[att="val"]	属性 att 值为"val"的 E 元素

代码实例：

```
p[id] {
  border: 5px solid black;
}
input[type='text'] {
  border: 5px solid red;
}
```

代码解析：

第一个选择器代码表示，为存在 id 属性的 p 元素，设置 5 像素的实线黑色边框；  
第二个选择器代码表示，为存在 type 属性，且属性值为 text 的 input 元素，设置 5 像素的实线红色边框。

### 11.5.3 模糊类属性选择器

三种“模糊类”查询的属性选择器如表 11.3 所示。



表 11.3 模糊类属性选择器

选择器语法	选择器含义
E[att ^ = "val"]	属性 att 的值以"val"开头的 E 元素
E[att \$ = "val"]	属性 att 的值以"val"结尾的 E 元素
E[att * = "val"]	属性 att 的值包含"val"字符串的 E 元素

代码实例：

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 属性选择器</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 500px;
      height: 40px;
      padding: 10px;
    }
    .wrap li {
      float: left;
      width: 36px;
      height: 36px;
      margin-right: 5px;
      border: 2px solid black;
      background: #ccc;
      color: #fff;
      text-align: center;
      line-height: 36px;
      font-size: 20px;
      border-radius: 20px;
    }
    /* E[att ^ = "val"]选择器 */
    .wrap li[id ^ = "n"] {
      background: #39f;
    }
    /* E[att $ = "val"]选择器 */
    .wrap li[id $ = "n"] {
      color: green;
    }
    /* E[att * = "val"]选择器 */
    .wrap li[id * = "n"] {
      border: 2px solid red;
    }
  </style>
</head>
<body>
```

```
<div class="wrap">
  <ul class="clearfix">
    <li id="one">1</li>
    <li id="two">2</li>
    <li id="three">3</li>
    <li id="four">4</li>
    <li id="five">5</li>
    <li id="six">6</li>
    <li id="seven">7</li>
    <li id="eight">8</li>
    <li id="nine">9</li>
    <li id="ten">10</li>
  </ul>
</div>
</body>
</html>
```

代码解析:

在类名为 wrap 元素中的 li 元素当中,为 id 属性以 n 开头的 li(第 9 个)设置浅蓝色背景;为 id 属性以 n 结尾的 li(第 7、10 个)设置绿色文字;为 id 属性包含 n 的 li(第 1、7、9、10)设置 2 像素的红色实线边框。

显示效果如图 11.10 所示。



图 11.10 属性选择器选择效果



## 11.6 表单元素的实际应用

### 11.6.1 去掉表单元素外部的聚焦线

在浏览器默认情况下,当表单中的各个表单元素被选中时,会有“聚焦状态”。以文本框为例,当聚焦时不同浏览器的显示效果如图 11.11 所示。

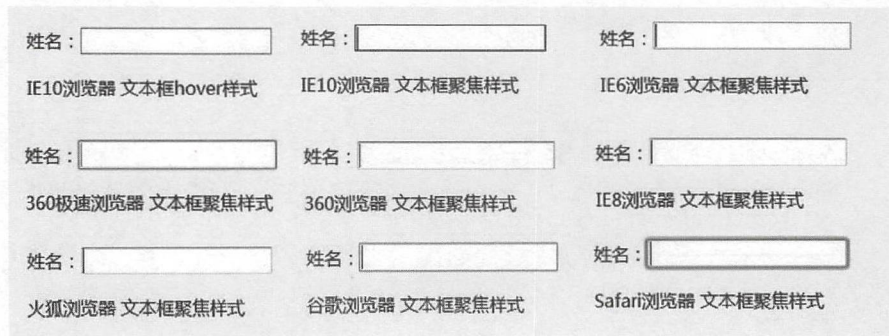


图 11.11 表单文本框



不同浏览器的默认聚焦状态并不相同(高亮部分的颜色、粗细),而开发方通常需要让各个浏览器的显示状态保持一致,因此,通常会去掉聚焦时的高亮部分。

为表单元素添加如下代码,即可取消掉聚焦后文本域外部的高亮显示。

```
outline: none;
```

## 11.6.2 textarea 的尺寸控制

默认的多行文本域的右下角,存在一个小三角,用户可以拖曳这个小三角进行多行文本框大小的调整。从网站的开发方考虑,并不希望用户进行多行文本域的大小操作(会导致页面布局发生变化甚至崩溃),此时就需要禁止用户的缩放。

为多行文本域添加如下代码即可去掉右下角拖动的小三角。

```
textarea {  
    resize: none;  
}
```

297

## 11.6.3 自定义样式的表单元素

### 1. 自定义样式表单元素的实现方法与原理

在实际的网页开发当中,对于表单元素,设计师通常会进行相应的设计,与默认状态下的表单元素显示效果可谓千差万别。要想实现一个“非默认状态”的表单元素,并不能够仅使用表单标签,而需要灵活地应用各类标签以及 CSS 样式的知识。通常称为“做假”,即运用各类标签以及表单标签,共同完成一个“自定义样式”的开发,其他标签主要是为了实现需要的样式,而表单标签主要是为了实现数据的提交,两者缺一不可。

### 2. 自定义样式表单实例详解

功能需求:实现如图 11.12 所示表单效果。

需求分析:

在效果当中,有两个地方需要用户输入文字,一种显示为正常文本,另一种为密码功能,因此需要选用文本框和密码框两种。

在视觉上来看,文本框和密码框的外部边框包含头像和密码锁两种图标,但是在实际当中,文本框和密码框都不存在所谓的图标,也无法为其定义图标,也就是说图标必须由其他元素来实现,而视觉上两个框外部的灰色边框,也绝对不属于文本域或密码框,需要额外使用其他标签来实现。

在文本域当中,有“请在此输入用户名”的提示性文本,在密码域当中,有“请在此输入密码”的提示性文本,此处不要理解为两个都是普通文本框,设计图在这里要传达出来的是:“两种文本域均存在提示性文本,而提示性文本需要单独以一个标签的形式而存在”。

整理一下如上的基本思路:第一,这是一个表单,需要使用到 form 标签;第二,包含两

图 11.12 自定义样式表单登录实例需求图

个输入框,分别是文本域和密码域;第三,存在两个元素用于处理输入框外部的边框;第四,两个小图标需要通过某个元素的背景图来设置;第五,每个输入框都存在一个提示性文本,需要额外增加元素来实现。

效果的基本结构:

```
<form class = "wrap" action = "">
  <div class = "user">
    <label for = "username">请输入用户名</label>
    <input type = "text" name = "username" id = "username">
  </div>
  <div class = "pass">
    <label for = "password">请输入密码</label>
    <input type = "password" name = "password" id = "password">
  </div>
</form>
```

标签层次关系如图 11.13 所示。

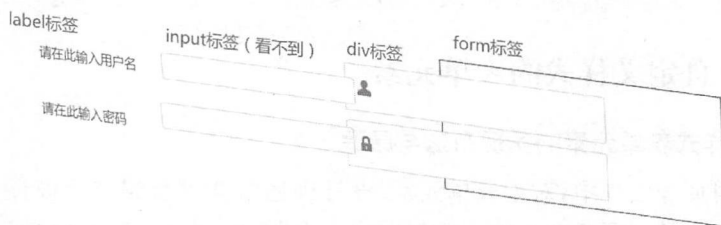


图 11.13 表单实例分析——分层结构

具体实现思路与代码:

input 和 label 均针对 div 元素定位,做成重叠效果。当单击 label 时,相对应的 input 元素就会被聚焦。

完整代码:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 表单实例</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 300px;
      height: 100px;
      padding: 12px;
      border: 1px solid #933;
    }
    .user, .pass {
      position: relative;
      height: 46px;
```



```

padding-left: 48px;
border: 1px solid #d9d9d9;
border-radius: 3px;
}
.user {
margin-bottom: 8px;
background: url('../images/user.png') 0 0 no-repeat;
}
.pass {
background: url('../images/pass.png') 0 0 no-repeat;
}
.wrap label, .wrap input {
position: absolute;
top: 0;
left: 48px;
}
.wrap label {
z-index: 1;
width: 250px;
height: 100%;
line-height: 46px;
color: #666;
cursor: text;
}
.wrap input {
width: 250px;
height: 100%;
border: none;
background: transparent;
line-height: 46px;
outline: none;
}
</style>
</head>
<body>
<form class="wrap" action="">
<div class="user">
<label id="userLabel" for="username">请输入用户名</label>
<input type="text" name="username" id="username">
</div>
<div class="pass">
<label id="passLabel" for="password">请输入密码</label>
<input type="password" name="password" id="password">
</div>
</form>
</body>
</html>

```

### 相关问题:

当鼠标单击 label 时,虽然能够让相对应的 input 聚焦,但是在输入内容时,label 并不会消失(或隐藏),此处单纯的 HTML 和 CSS 没有办法实现这个功能,需要使用 JavaScript 来辅助。(在上面这段代码中 label 的 id 名也是为了 JS 处理而存在。)

在代码当中使用到了 border-radius,表示圆角,是第 13 章要讲解的内容,不兼容 IE8-。如果希望兼容 IE8-,则需要将整个圆角矩形截图,作为背景图设置,也可以引入 JS 插件。背景图的方式在此就不额外书写代码了。

具体 JavaScript 范例代码如下。

```
<script>
    var userText = document.getElementById('username');
    var userLabel = document.getElementById('userLabel');
    var passText = document.getElementById('password');
    var passLabel = document.getElementById('passLabel');
    userText.onkeyup = function() {
        changeLabel(this, userLabel)
    }
    passText.onkeyup = function(){
        changeLabel(this, passLabel)
    }
    function changeLabel(valEle, labelEle) {
        var val = valEle.value.split(" ").join("");
        if (val == "") {
            labelEle.style.display = "block";
            valEle.value = "";
        } else {
            labelEle.style.display = "none";
        }
    }
</script>
```



## 第12章

# 停下来回头看路



### 12.1 从○开始

#### 12.1.1 ○是什么

本章将借助这个“○”的游戏,来讲解一下复习时容易陷入的问题,以及如何合理地进行复习。

游戏要求:请充分发挥你的想象,看看“○”可以是什么(例如:呼啦圈)?

操作要求:使用一张白纸,将答案用笔书写下来。

操作人数:1~5人(多人参与时,只允许一人执笔,4或5人团队式合作构思,游戏效果最佳)。

时间限制:5分钟。

#### 12.1.2 最常见的答案

(1)可能这和你的答案很相似:

足球,太阳,圆环,数字零,月亮,乒乓球,排球,月饼,灯泡,钥匙环,方向盘,表盘,葡萄,苹果,水滴,馒头,UFO,字母o,数字8,电风扇……(在此不一一罗列)

(2)看上去“最为常见”的答案,拥有什么问题?

① 在思考的时候,你的思维方式是否有规律可循?

② 结果当中,很有可能同时拥有“排球”、“球类”这两种不同“级别层次”的答案。

③ 结果当中,通常答案没有按照分类书写到一起,如上的“数字零”、“数字8”。如果是这种现象,说明在思考的时候思维比较跳跃。跳跃性的思维虽然能够拓宽“面”,却很容易丢掉一些东西。

④ 结果当中,有可能会出现重复,游戏的人数越多,这种重复的概率也就越高。

#### 12.1.3 让结果变得更优秀

1. 合理地运用发散思维和聚合思维

如何让“游戏结果”变得优秀起来呢?

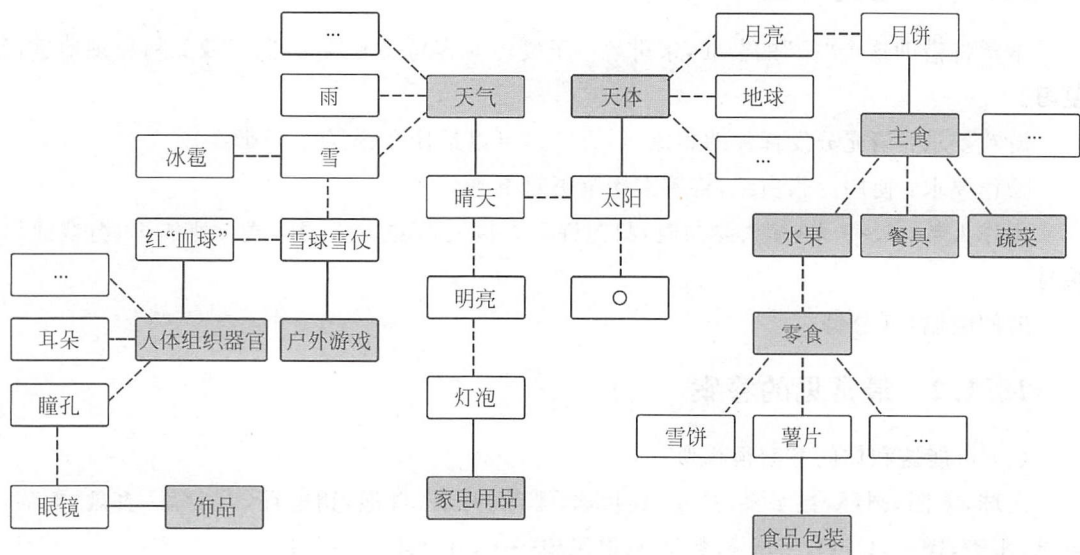
此前我们的思维方式,每次的答案通常都是从○出发,进行构思,思考○可能是什么。  
每次思考到其他方面的东西之后,可能并没有进行深入思考(总结),也没有进行一些“联想”。

学习知识也是如此,头脑中的知识,可以是一个一个的散点,也可以是一个网络或者一个具有很强组织结构性的知识体系。因此,复习方式直接影响着知识的掌握深度和熟练度。

无论是在这场游戏当中,还是复习当中,有两种思维模式必不可少,一种是“发散思维”,即基于当前进行联想,让自己想到更多;另一种是“聚合思维”,即把当前想到的内容进行归纳总结,找到相似的内容。

2. 构思范例

以○为例,看看发散思维与聚合思维在游戏当中的应用。  
不断地发散(联想)、聚合(总结),交替进行操作,如图 12.1 所示。



备注: 由于区域有限, 此处并没有将每个分类和具体项目罗列出来  
---- 虚线表示发散思维  
—— 实线表示聚合思维

图 12.1 ○是什么

完成整体的发散和思考之后,需要进行总结和整理。可以将上面思考的结果罗列成如下格式。

- 天体: 太阳、月亮、卫星轨道、……
- 天气: 水滴(雨点)、冰雹、雪粒、……
- 主食: 馒头、月饼、汤圆(元宵)、馅饼、……
- 水果: 苹果、西瓜、樱桃、……
- 餐具: 碗、盘、汤匙(勺)、……
- 蔬菜: 西红柿、圆茄子、洋白菜、生菜球、……



人体组织器官：头部、眼睛、红细胞、……

户外游戏：足球、篮球、滚铁环、呼啦圈、毽子的底部、……

饰品：手表的表盘、手镯、圆的眼镜镜框、……

家电用品：手电筒、电灯泡、电饭锅、电风扇、……

零食：雪饼、薯片、麦丽素、果冻、……

食品包装：桶装盒、(KFC)全家桶、盛汤的快餐盒、……

小结：

通过这种方法进行思考得到的答案，逻辑性会很强，同时并不会出现重复性的内容书写。在构思过程中，通过联想的方式，把一个一个的知识点连接成了一条一条的线。

对于 HTML5 知识的学习，也是这样一个道理。在前面十几章的讲解当中，是以“网站开发”为主线进行推进，随着网站开发的深入，逐渐地接触到相关的知识，这种方式是最贴近于实践的方式，也是掌握起来比较容易的方式。但是通过这种方式复习时，知识会比较零散（例如 CSS 选择器，就分布于 4 个章节当中），并不适合按照原有方式进行“复习”。

因此需要用不同于学习的方式，来进行复习。复习的方法主要有两种，一种方法是充分利用“发散思维”的网状复习法，这种方法主要是依靠知识之间的相似特性，建立知识和知识之间的联系，从而让知识从点变成网，降低知识的遗忘速度；另一种方法是充分利用“聚合思维”的归纳整理法，这种方法主要是将各个知识点分类，进行整理总结，从而让知识变得体系化。



## 12.2 网状复习法

### 12.2.1 网状复习法的特点

网状复习法，主要应用“发散”思维，利用已有知识联系新知识，完成知识网络化。

我们认为知识并不是一个个的散点，不是孤立存在的，每个知识与每个知识之间都存在着一定的联系，在学习知识时，尽可能多地创造知识之间的联系，让知识记得更牢固，得到更好的理解。

优势：更容易建立知识之间的联系，通过该方法复习，需要针对每个知识点深入思考，因此能够加强对知识应用方面的理解；由于建立了知识之间的联系，记忆的知识也不容易遗忘。

劣势：复习时容易造成部分知识的遗漏，可借助介绍的第二种方法（归纳总结）弥补这个劣势。

### 12.2.2 网状复习法的实现方式

建议使用白纸和笔进行操作，不建议使用计算机。

从某一个知识点开始书写，之后尝试从这个知识点的某一个特性出发，思考出其他有关联的知识点，然后以此类推，尽可能地将多个知识“连接”在一起，在每两个知识的连接线之间标注出关联的原因。

注意：具有联系的两个知识点，可以是同级别的知识点，也可以是一个概括性的知识点和一个具体的知识点，在这一方面并没有限制。

虽然每次建立的网络可能都有所不同，但是，经过一次次的书写和“连接”，能够为知识逐渐建立联系，让知识从一个个散点成为一个网络。

### 12.2.3 网状复习法的简单案例

HTML 与 CSS 的网状复习法如图 12.2 所示。

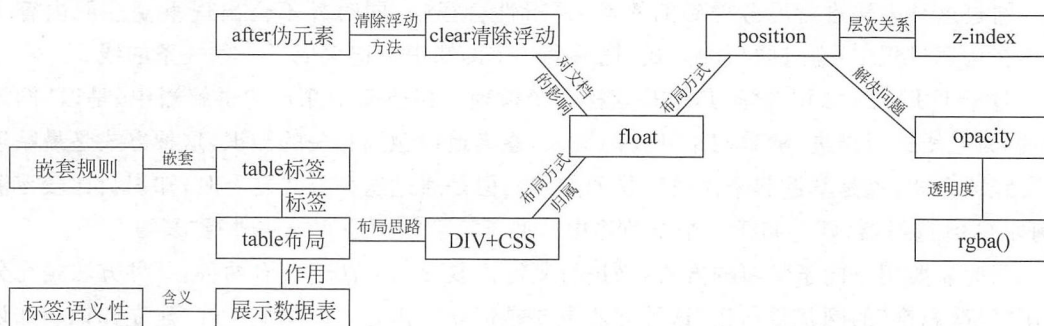


图 12.2 HTML 与 CSS 的网状复习法

案例解析：

在此处以 float 为起点，先使用“发散”思维思考 float 具备的基本特点：

- (1) float 属于浮动布局；
- (2) float 是当前 HTML+CSS(或 DIV+CSS)布局的一种具体形式；
- (3) float 会让文档脱离文档流；
- (4) float 能够让块或行元素，既能够设置宽高，又能够和其他浮动元素处于同一行。

由以上 4 个基本特点，分别能够想到：

- (1) position 也是一种布局方式；
- (2) DIV+CSS 布局包含 float 布局；
- (3) 浮动之后需要进行浮动的清除，从而防止文档布局受到影响；
- (4) 块元素与行元素(这个点在图 12.2 中没有绘制)。

(当然有可能你会想到其他的特点和相关知识)。

在构思完成 float 的相关技术之外，再基于某一个分点(比如“DIV+CSS”)做细化思考。每个点都遵循 float 的构思模式，不断地将图变大。

当整个知识体系建立完成之后，可以查看每个点，然后回忆具体知识点。

### 12.2.4 网状复习 HTML 与 CSS

在此给出网状复习的实例，如图 12.3 所示。需要注意的是：这只是网状复习的一种思路，完全可以通过别的关系建立两种知识之间的联系。另外，实例当中并没有涉及所有的知识点，该实例仅作参考。(该实例是从“src、href”作为起点进行思考的。)



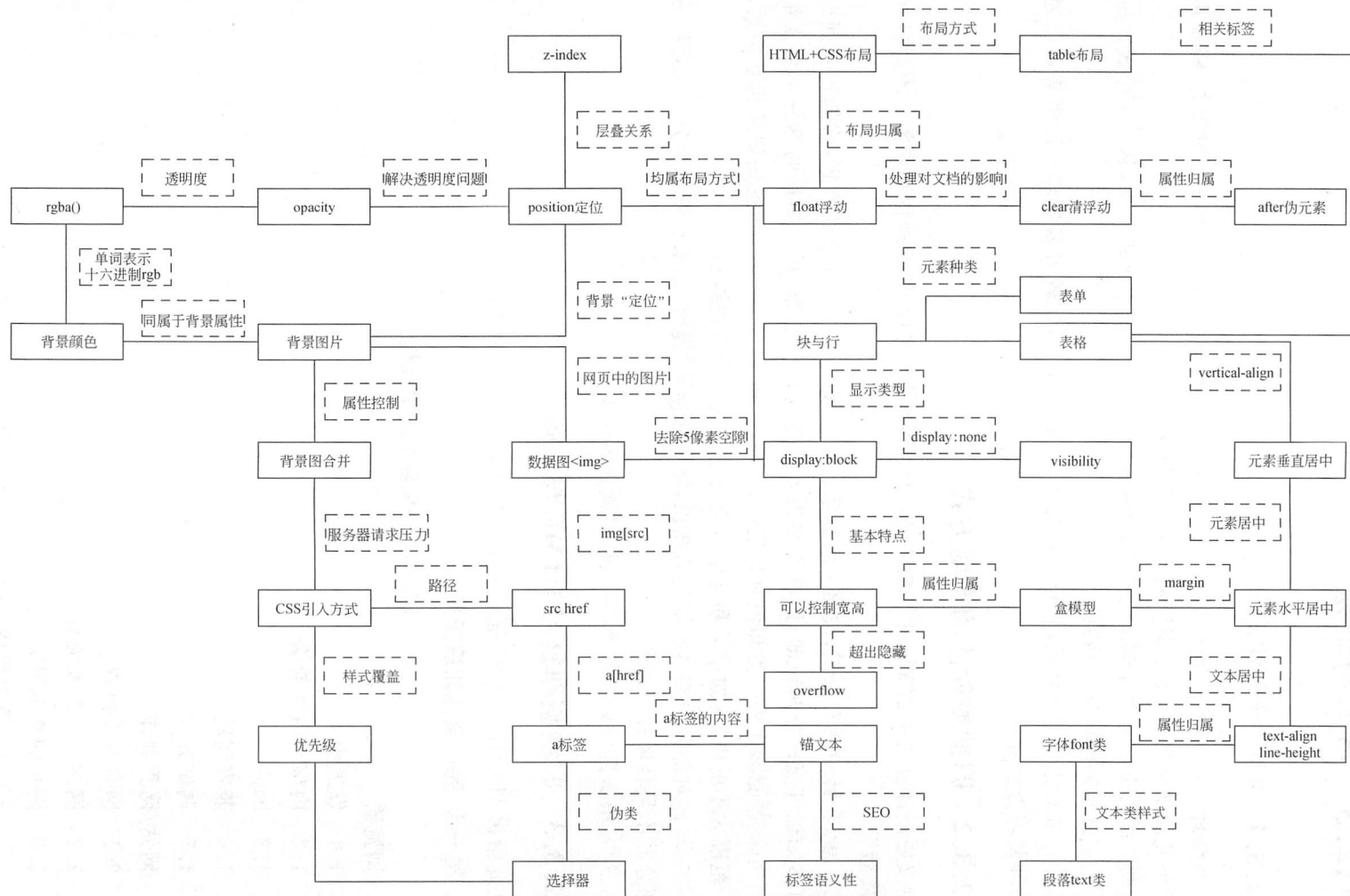


图 12.3 HTML 与 CSS 的网状复习法案例



## 12.3 归纳整理法

### 12.3.1 归纳整理法的特点

归纳整理,主要应用“聚合思维”,将学过的知识点集中起来,按照一定的分类,将知识“放置”到几层结构当中,使知识形成包含与被包含的关系。

优势:能够全面地覆盖到所有知识,清晰地表达出知识的上下级关系,从一个个凌乱的知识变成逻辑清晰的知识体系。

劣势:知识之间缺乏联系,相对比较独立,不利于应用,比较容易遗忘。该劣势可以通过第一种复习方法弥补。

### 12.3.2 归纳整理法的实现方式

建议使用计算机中的 Word 等软件,使用自动编号来实现,便于操作和调整。不建议使用“纸质版”。

第一步,先针对所有知识,按照一级标题、二级标题、三级小标题的方式,将知识罗列出来。有些知识可能拆分两级即可,有些知识可能需要拆分三级,每个知识的拆分不要太细化,每个小标题能够代表一类知识时为最佳。在这个过程当中,需要注意知识的相互关系,同一个范围大小的内容(如浮动和定位),应当处于同一个标题级别。

第二步,针对部分能够再次细化的知识,将单独标题“提取”出来再进行拆分。注意拆分后的部分并不是具体知识细节。

第三步,根据拆分后的每个小标题,回顾具体知识点。

### 12.3.3 归纳整理复习 HTML 与 CSS

在此给出 HTML 与 CSS 部分的总结归纳内容,仅供参考。

#### 1. 第一步 完成整体知识架构

##### HTML、CSS 整体知识架构

#### 1 浏览器

- 1.1 发展史
- 1.2 内核与内核前缀
- 1.3 hack
- 1.4 兼容问题
- 1.5 调试方法

#### 2 网站/网页制作

- 2.1 网页开发流程
- 2.2 网站制作开发流程
- 2.3 网页的基本组成
  - 2.3.1 语言组成(HTML+CSS+JS)
  - 2.3.2 结构组成(head、body)



- 2.3.3 编码格式
  - 2.3.4 文档声明
  - 2.3.5 元信息
- 2.4 布局
  - 2.4.1 定位布局
  - 2.4.2 浮动布局
  - 2.4.3 其他(table、DIV+CSS 区别)
- 3 标签的基本知识
  - 3.1 种类
    - 3.1.1 块元素
    - 3.1.2 行元素
    - 3.1.3 第三类元素
  - 3.2 语义性 & 选用
  - 3.3 书写规范
    - 3.3.1 嵌套
    - 3.3.2 缩进
  - 3.4 SEO 相关
  - 3.5 注释书写方法
- 4 样式
  - 4.1 CSS 引入方式
  - 4.2 CSS 选择器
  - 4.3 CSS 显示属性
  - 4.4 CSS 自身属性(盒模型)
  - 4.5 CSS 文本属性
    - 4.5.1 text
    - 4.5.2 font
    - 4.5.3 background
  - 4.6 代码规范
    - 4.6.1 命名
    - 4.6.2 书写顺序
    - 4.6.3 基本规范(空格等)
- 5 工具及其他相关技术
  - 5.1 背景图合并
  - 5.2 CSS Sprite
  - 5.3 Photoshop 抠图(只针对 PSD)

## 2. 第二步 细化具体每个小标题

在第一步当中,标签的基本知识与样式部分,需要进行细化,具体内容细化如下。

### 1 块元素

#### 1.1 div 布局作用



- 1.2 ul、ol、dl 使用
- 1.3 h 系列标签
- 1.4 p 段落标签
- 1.5 备注：特殊 iframe
- 2 行元素
  - 2.1 span
  - 2.2 strong、em、b、i 辨析
  - 2.3 img
    - 2.3.1 图片路径的详解
    - 2.3.2 alt、title 含义
    - 2.3.3 border
    - 2.3.4 5px 兼容问题
  - 2.4 a 链接与 title 含义
- 3 表格
  - 3.1 基本的嵌套规则
  - 3.2 caption、thead、tbody、tfoot、tr、td、th(基本组成)
  - 3.3 表格专有属性介绍
  - 3.4 表格布局与 DIV+CSS 布局的辨析
  - 3.5 利用表格进行页面的制作
- 4 表单
  - 4.1 input 系列
    - 4.1.1 文本框
    - 4.1.2 单选框
    - 4.1.3 多选框
    - 4.1.4 密码框
  - 4.2 select、option 下拉菜单
  - 4.3 textarea 文本域、label
  - 4.4 新增的表单元素
  - 4.5 制作基本表单的页面
- 5 HTML5 新增标签
  - 5.1 注：关于如何让 IE9-兼容放置在 JS 部分中讲解
- 6 CSS 引入方式及选择器
  - 6.1 三种引入方式及使用范围、条件
  - 6.2 三种基本选择器
  - 6.3 后代、群组
  - 6.4 属性选择器、通配符选择器、子选择器
  - 6.5 伪类选择器
- 7 伪元素
  - 7.1 :after、:before 的作用





## 7.2 清除浮动的应用

## 7.3 处理空标签

# 8 CSS 显示属性

## 8.1 display 显示类型

### 8.1.1 块与行的区别

### 8.1.2 display:none 的含义

### 8.1.3 display:inline-block 与浮动的不同视觉效果; display:table 等特殊显示方式及兼容问题

### 8.1.4 利用 display 处理垂直居中

## 8.2 float

### 8.2.1 文档流, float 对显示类型的影响

### 8.2.2 浮动的原理

### 8.2.3 浮动后对文档的影响

### 8.2.4 如何清除浮动

## 8.3 position

### 8.3.1 定位对文档的影响

### 8.3.2 伪 3D 及叠层, 父子级的 z-index 计算方法

### 8.3.3 定位的基本方法和原理

## 8.4 overflow

### 8.4.1 后台数据的要求

### 8.4.2 溢出的省略号/隐藏

### 8.4.3 overflow:hidden & overflow:auto 的区别

## 8.5 visibility:hidden 与 display:none 的区别

# 9 自身属性

## 9.1 如何理解盒模型

## 9.2 外边距与内边距的使用习惯

## 9.3 外边距存在的一系列兼容问题

## 9.4 布局的基本思想

## 9.5 缩写方式(border 的特殊缩写的方式)

## 9.6 最大/小宽度/高度

## 9.7 百分比的使用(过渡到 CSS3 响应式)

# 10 文本属性 text

## 10.1 文本水平垂直居中 text-align、vertical-align

## 10.2 垂直居中 line-height 行高(单行文本垂直居中)

## 10.3 文本修饰 text-decoration

## 10.4 首行缩进 text-indent

## 10.5 word-spacing、letter-spacing

# 11 文本属性 font

## 11.1 颜色 color



## 11.1.1 RGB 表示法(RGBA)

## 11.1.2 单词

## 11.1.3 十六进制

## 11.2 透明度 opacity(兼容问题)

## 11.3 font-family

## 11.3.1 常见字体种类

## 11.3.2 宋体,微软雅黑的英文书写方法

## 11.4 font-size

## 11.4.1 字体大小(px 与 em)

## 11.4.2 网页默认字体大小

## 11.5 font-weight、font-style

**12 文本属性 background(背景)**

## 12.1 背景色与背景图

## 12.2 背景定位与滚动、重复

## 12.3 背景图与 img 图的选择

## 12.4 背景图合并

**13 SEO**

## 13.1 语义性

## 13.2 代码规范

## 13.3 title 及 meta

## 13.4 关键字,a 标签的锚文本

## 13.5 白帽、黑帽优化

**3. 第三步 根据每个小标题进行知识回顾**

在头脑中根据小标题(知识的“关键词”)进行具体知识内容的回顾,如果记忆模糊,及时进行复习即可,此处不再具体书写每个小标题的知识点。

**12.4 hack 技术****12.4.1 什么是 hack 技术**

行业中存在着各类浏览器,典型的浏览器包括 IE、火狐、谷歌等;同样种类的浏览器,也存在着不同的版本,如 IE6、IE7、……、IE11 等。不同浏览器对于 CSS 的解析机制并不是完全相同的,因此有时会导致不同浏览器当中,页面的效果各不相同,没有办法达到想要的“统一”效果。

此时可以针对某种浏览器进行样式的设置,从而达到所有浏览器中显示效果的一致性,这种能够标识出不同浏览器的书写方式就是 hack。

简单地讲就是可以通过 hack 技术,只针对某一种或几种浏览器进行样式设置。





## 12.4.2 常用 IE hack

常用 IE hack 技术如表 12.1 所示。

表 12.1 常用 IE hack 技术

代码范例	针对的浏览器	描述
<code>_width: 400px;</code>	IE6	将宽度设置为 400 像素
<code>+width: 300px;</code>	IE6、IE7	将宽度设置为 300 像素
<code>* width: 400px;</code>	IE6、IE7	将宽度设置为 400 像素
<code>width: 200px\9;</code>	IE6~IE10	将宽度设置为 200 像素
<code>width: 100px\0;</code>	IE8~IE11	将宽度设置为 100 像素

备注：该 hack 的测试时间为 2016 年 09 月底。

## 12.4.3 IE 条件注释

IE 中的条件注释(Conditional Comments)对 IE 的版本和 IE 非 IE 有优秀的区分能力，是 Web 开发中常用的 hack 方法，能对 IE 系列产品进行单独的 HTML 代码处理。

### 1. 基本语法

```
<!-- [if 属性 IE 版本号 ]>
    具体 HTML 代码
<![endif] -->
```

### 2. 语法解析

条件注释的书写方法与 HTML 的注释(<!-- -->)相同，IE 浏览器将会根据 if 条件来判断是否解析条件注释里的内容，而 IE 以外的浏览器将会把它们看作是普通的注释而忽略。

条件注释能够应用于 IE5 以上的各类 IE 版本浏览器，如果用户安装了多个 IE 浏览器，条件注释将以最高版本的 IE 浏览器为标准。

### 3. 基本属性

gt: greater than, 选择条件以上版本，不包含条件版本。

lt: less than, 选择条件以下版本，不包含条件版本。

gte: greater than or equal, 选择条件以上版本，包含条件版本。

lte: less than or equal, 选择条件以下版本，包含条件版本。

!: 选择条件版本以外所有版本，无论高低。

### 4. 代码实例

```
<!-- [if IE 8]>
    <link rel="stylesheet" href="css/basic.css">
<![endif] -->
<!-- [if gt IE 8]>
```



```
<link rel = "stylesheet" href = "css/demo.css">
<![endif] -->
```

## 5. 代码解析

只为 IE8 浏览器加载 basic.css 文件；为 IE8 以上的浏览器（不包含 IE8），加载 demo.css。



## 12.5 实现网页开发之后要考虑的东西

网页开发完毕之后，并不意味着开发工程师工作的结束，对于开发工程师来说，还需要进行一系列的优化，其中最基本的几项包括：CSS 代码压缩、JavaScript 代码压缩、图片压缩、ico 文件的制作以及 404 页面的制作。

CSS、JS 代码压缩工具：可以直接在搜索引擎中查找“CSS/JS 代码格式化工具”，也可以为 Sublime Text 安装相应插件。

图片的压缩：可以手动进行压缩，也可以通过在线网站完成（图片压缩网址：<http://zhitu.isux.us/>或 <https://tinypng.com/>），压缩的原则是“在不影响图片视觉质量的前提下，尽可能的小，可以反复压缩”。

ico 文件：ico 文件是显示于浏览器网页选项卡中，title 前面的小图标，部分浏览器需要在服务器环境中才会显示。ico 文件的制作，也可以通过在线生成工具完成，之后通过如下命令，为网页添加 ico 文件即可。（ico 文件在线生成网址：<http://www.faviconico.org/>。）

```
<link rel = "shortcut icon" href = "favicon.ico">
```



## 12.6 PC 端浏览器的兼容问题

随着 IE6 浏览器的退市，PC 端浏览器的兼容问题逐渐成为过去。在此，不再针对 PC 端浏览器兼容进行讲解，如果希望了解 IE6 浏览器的渲染方式，以及常见的 IE6 浏览器兼容问题，可以进入 HTML5 学堂官网进行查看。





# 第13章

## HTML5新标签与CSS3基础



### 13.1 HTML5 新增元素

HTML 从第 4 版本向第 5 版本“升级”的过程中,新添加了不少标签和属性。在这些标签当中,并不是所有的标签都会在开发中频繁使用,在学习时,建议先掌握新增结构标签,在现代浏览器中能够合理应用,之后再研究视音频、画布等功能类新增标签。

#### 13.1.1 新增结构元素及含义

##### 1. 新增结构元素

<code>&lt;header&gt;</code>	定义页眉
<code>&lt;hgroup&gt;</code>	定义对网页标题的组合
<code>&lt;nav&gt;</code>	定义导航
<code>&lt;section&gt;</code>	定义文档中的区段
<code>&lt;time&gt;</code>	定义日期和时间
<code>&lt;article&gt;</code>	定义文章
<code>&lt;aside&gt;</code>	定义文章的侧边栏
<code>&lt;figure&gt;</code>	定义一组媒体对象以及文字
<code>&lt;figcaption&gt;</code>	定义 figure 的标题
<code>&lt;footer&gt;</code>	定义页脚

header 与 footer 标签用于表示页面、应用或某个模块的头部和尾部; nav 用于表示导航; article 标签表示文章; aside 标签和 article 标签的用法类似,只是表示内容会出现在侧边栏; figure 标签表示一段自包含的内容——独立的流内容(图像、图表、照片、代码等)。

在开发当中,可以使用 section 和 article 标签替代过度使用的 div,用这些标签来对页面进行内容划分。

##### 2. 其他新增元素

<code>&lt;video&gt;</code>	定义视频
<code>&lt;audio&gt;</code>	定义音频
<code>&lt;embed&gt;</code>	插入各种多媒体
<code>&lt;mark&gt;</code>	定义需要突出显示或高亮显示的文本



<progress>	显示 JS 中耗费的函数的进程
<time>	表示时间或日期
<canvas>	定义图形,提供画布
<command>	表示命令按钮
<details>	表示用户要求得到并可以得到的详细信息
<wbr>	表示软换行

### 13.1.2 使用 HTML5 新结构元素完成页面搭建

#### 1. 功能需求

图 13.1 是在日常各类网站中比较常见的页面结构,请思考:如何使用 div 实现基本结构?

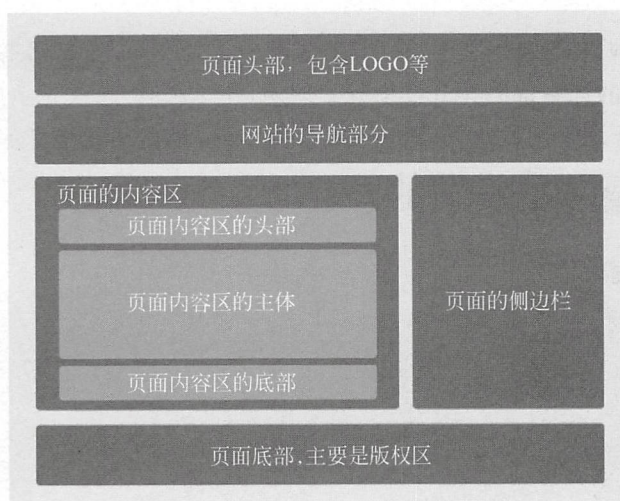


图 13.1 页面结构搭建功能需求

#### 2. 使用 div 实现基本结构

代码实例:

```
<div class = "wrap">
  <div class = "header"></div>
  <div class = "nav"></div>
  <div class = "con">
    <div class = "con - head"></div>
    <div class = "arc"></div>
    <div class = "con - foot"></div>
  </div>
  <div class = "sidebar"></div>
  <div class = "footer"></div>
</div>
```





结构搭建如图 13.2 所示。

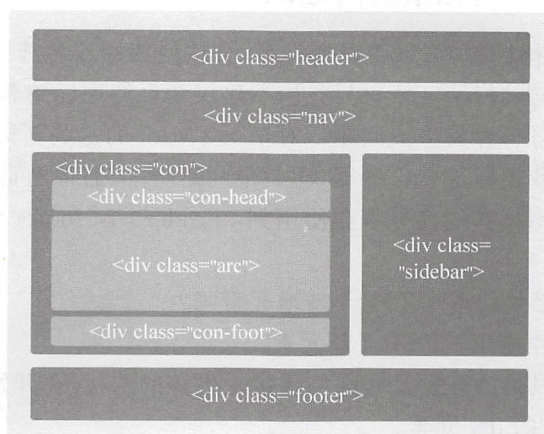


图 13.2 div 实现页面结构搭建

### 3. 使用新增结构标签实现基本结构

使用 HTML5 新增结构元素配合 div, 替换掉原有“纯 div”的基本结构。

代码实例：

```
<div class = "wrap">
  <header></header>
  <nav></nav>
  <section class = "con">
    <header></header>
    <article></article>
    <footer></footer>
  </section>
  <aside></aside>
  <footer></footer>
</div>
```

结构搭建如图 13.3 所示。

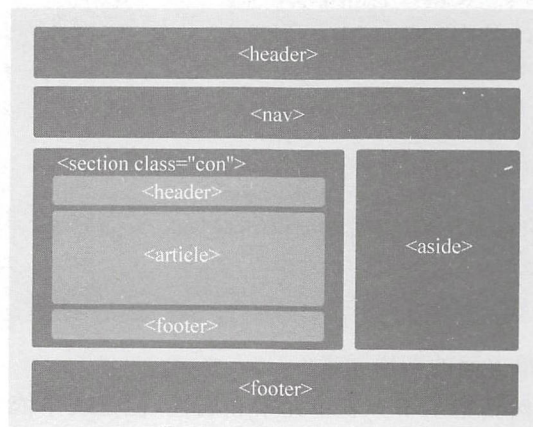


图 13.3 新标签实现页面结构搭建



### 13.1.3 HTML5 新元素的问题区

(1) 如果使用新结构标签,可以不设置类名,直接使用标签名选择器吗?

在刚开始接触选择器时(第3章)提到过,不建议新手使用标签名选择器,主要是由于标签名选择器选择范围过广,容易造成问题。对于HTML5的这些新标签,也是同样的道理。

当然,可以适当地使用标签名选择器,在一个网站当中,通常网页的头部、底部和导航都是相同的,如果确定在网站其他部分不会使用到<header>、<footer>、<nav>,就能够使用标签名选择器。

(2) 在开发当中,必须使用HTML5新增元素吗?

在开发中,并非必须使用HTML5新元素。

HTML5新元素,只是为开发工程师们提供了更多的选择,在开发时适当、合理地使用,能够减少代码中“大量”的div元素。

只要合理开发,是否使用新元素并不重要。但是,由于新元素比单纯的div元素,更有助于结构的搭建,建议开发者在主流浏览器中使用。

(3) 如何让低端浏览器兼容新的HTML5元素?

在IE8-低端浏览器当中,并不存在header、footer等新增标签,所以需要使用JavaScript动态地创建结构元素,之后使用CSS,为每个结构元素赋以“块元素”的显示特点。

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 兼容HTML5 新元素</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    header, footer, nav, section, aside, details {
      display: block;
    }
  </style>
  <script>
    var arr = ['header', 'footer', 'nav', 'section', 'aside', 'details'];
    for (var i = 0; i < arr.length; i++) {
      document.createElement(arr[i]);
    };
  </script>
</head>
<body>
  <header>此时,在IE8-下可以正常显示</header>
  <footer>此时,在IE8-下可以正常显示</footer>
</body>
</html>
```

(4) 为何不讲解表单方面的新增元素?

HTML5推出时虽然新增了不少表单元素和属性,但是表单依旧表现出了严重的兼容问题。出于兼容情况的考虑,在实际开发当中,通常都是通过表单做假或者插件来解决表单



问题,新增的表单元素就成了鸡肋。

在新增的表单元素、属性当中,还是有三个属性需要了解一下,这三个属性分别是:placeholder 占位符;autofocus 自动聚焦、hidden 隐藏(相当于是让元素 display: none,不显示在页面当中)。

关于 placeholder 属性的代码实例:

```
<input type="text" name="" id="" placeholder="HTML5 布局之路">
```

显示效果如图 13.4 所示。



图 13.4 placeholder 属性实现效果

控制表单元素 placeholder 属性中文字的颜色:

```
input:-moz-placeholder {  
    color: #39f;           /* 火狐 4~18 */  
}  
input::-moz-placeholder {  
    color: #39f;           /* 火狐 19+ */  
}  
input:-ms-input-placeholder {  
    color: #39f;           /* Trident 内核(IE) */  
}  
input::-webkit-input-placeholder {  
    color: #39f;           /* webkit 内核(谷歌等) */  
}
```

如果希望了解表单的新增元素,可以查看本书提供的电子案例中的“input 元素的新类型.html”和“input 元素的新属性.html”文件,在此就不再详细讲解了。



## 13.2 浏览器内核

### 13.2.1 浏览器主要内核

#### 1. 什么是浏览器内核

浏览器最核心的部分是“Rendering Engine”,可译为“渲染引擎”,不过也有习惯将其称为“浏览器内核”。渲染引擎的作用是将代码“变”成网页。详细来说,就是取得网页的代码、文字、图像等内容,之后进行信息整理,并计算网页的显示方式,最后显示在显示器的浏览器当中。

#### 2. 为何要了解浏览器内核

不同的浏览器内核对网页编写语法的解释也有所不同,因此同一网页在不同内核的浏览器里的渲染(显示)效果也可能不同,这也是前端工程师需要了解内核的原因。前端工程

师在代码中通过一些内核前缀为相应浏览器增加某些功能。

3. 常见浏览器内核

常见浏览器内核如表 13.1 所示。

表 13.1 常见浏览器与浏览器内核

浏 览 器	最新的浏览器内核	备 注
IE 浏览器	Trident	
谷歌浏览器 Chrome	Blink	旧版本内核为 WebKit
苹果浏览器 Safari	WebKit	
火狐浏览器 FireFox	Gecko	
欧朋浏览器 Opera	WebKit	旧版本为 Presto 内核

4. 360 浏览器

360 安全浏览器是一款双内核的浏览器,极速模式使用的是 WebKit 内核,WebKit 内核是全球最快速的浏览器内核,同时支持了诸多的网页新标准;兼容模式使用的是 IE 浏览器所使用的 Trident 内核。

让 360 浏览器强制以 WebKit 内核渲染的方法,是在< head>标签中添加如下代码:

```
<meta name = "renderer" content = "webkit">
```

13.2.2 常见浏览器内核前缀

每种浏览器内核,都对应着浏览器内核前缀,如表 13.2 所示。浏览器内核前缀,在书写 CSS 代码时会使用到,对于添加前缀的 CSS 样式代码,只有相应内核的浏览器才能够解读该代码。

表 13.2 浏览器内核与内核前缀对照表

浏览器内核	浏览器内核前缀	浏览器内核	浏览器内核前缀
Trident	-ms	Gecko	-moz
Blink	-webkit	Presto	-o
WebKit	-webkit		

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 不同内核前缀书写样式</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 200px;
      height: 200px;
```



```

        border: 5px solid black;
        -webkit-border-radius: 5px;
        -moz-border-radius: 10px;
    }
</style>
</head>
<body>
    <div class = "wrap"></div>
</body>
</html>

```

代码解析：

为 WebKit 内核的浏览器设置 5 像素的圆角边框；

为 Gecko 内核的浏览器设置 10 像素的圆角边框；

其他内核的浏览器没有圆角边框。

### 13.2.3 浏览器内核的问题区

是不是所有的样式都可以使用内核前缀？

并不是所有的样式都可以设置内核前缀。

之所以存在内核前缀，是由于 CSS3 最初发布时，并非所有的浏览器都支持，为了互相竞争，各自浏览器推出了以自己内核前缀开头的样式，之后，随着样式的普及，各个浏览器才都支持 W3C 组织规定的样式，对于 CSS3 之前的大部分样式属性，并没有内核前缀的说法。

如下代码实例中，设置的样式代码并不会生效。

```

- ms - height: 200px;
- moz - height: 300px;
- webkit - height: 400px;

```



## 13.3 CSS3 选择器

### 13.3.1 CSS2.0 选择器回顾

在 CSS2.0 版本当中，共接触到了 10 种选择器，分别是第 3 章中的“ID 选择器”、“类名选择器”、“标签名选择器”；第 5 章中的“群组选择器”、“后代选择器”、“子代选择器”、“通配符选择器”、“毗邻选择器”；第 6 章中的“伪类选择器”和第 11 章中的“属性选择器”。

### 13.3.2 CSS3 选择器——通用兄弟选择器

基本语法：

```
E ~ F { /* 样式代码 */ }
```

代码解析：

匹配任何在 E 元素之后的同级 F 元素。

通用兄弟元素选择器是 CSS3 新增加的一种选择器,这种选择器将选择某元素后面的所有兄弟元素,与毗邻选择器类似,需要在同一个父元素之中。

代码实例:

```
p ~ ul { background: # ff0; }
```

13.3.3 CSS3 选择器——伪类选择器

1. 结构性伪类

部分结构类伪类选择器如表 13.3 所示。

表 13.3 结构类伪类选择器 01

选择器语法	选择器含义
E:root	匹配文档的根元素,对于 HTML 文档,就是 HTML 元素
E:nth-child(n)	匹配其父元素的第 <i>n</i> 个子元素,第一个编号为 1
E:nth-last-child(n)	匹配其父元素的倒数第 <i>n</i> 个子元素,第一个编号为 1
E:last-child	匹配父元素的最后一个子元素,等同于:nth-last-child(1)

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - CSS3 结构伪类选择器 1</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 500px;
      height: 40px;
      padding: 10px;
    }
    .wrap li {
      float: left;
      width: 36px;
      height: 36px;
      margin-right: 5px;
      border: 2px solid black;
      background: # ccc;
      color: # fff;
      text-align: center;
      line-height: 36px;
      font-size: 20px;
      border-radius: 20px;
    }
  /* nth-child(n)选择器 */
```



```
li:nth-child(1) {
    background: #39f;
}
/* nth-last-child(n)选择器 */
li:nth-last-child(1) {
    color: green;
}
/* nth-child(2n)选择器 */
li:nth-child(2n) {
    border: 2px solid red;
}
</style>
</head>
<body>
    <div class = "wrap">
        <ul class = "clearfix">
            <li>1</li>
            <li>2</li>
            <li>3</li>
            <li>4</li>
            <li>5</li>
            <li>6</li>
            <li>7</li>
            <li>8</li>
            <li>9</li>
            <li>10</li>
        </ul>
    </div>
</body>
</html>
```

代码解析：

为第一个 li 元素添加蓝色的背景；为最后一个 li 元素设置文字颜色为绿色；为所有的偶数的 li 设置红色边框。

显示效果如图 13.5 所示。



图 13.5 CSS3 结构类伪类选择器选择效果 01

部分结构类伪类选择器如表 13.4 所示。

表 13.4 结构类伪类选择器 02

选择器语法	选择器含义
E:nth-of-type(n)	与:nth-child()作用类似,但是仅匹配使用同种标签的元素
E:nth-last-of-type(n)	与:nth-last-child()作用类似,但是仅匹配使用同种标签的元素
E:first-of-type	匹配父元素下使用同种标签的第一个子元素,等同于:nth-of-type(1)
E:last-of-type	匹配父元素下使用同种标签的最后一个子元素,等同于:nth-last-of-type(1)

## 代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - CSS3 结构伪类选择器 2 </title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 500px;
      height: 40px;
      padding: 10px;
    }
    .wrap dd, .wrap dt {
      float: left;
      width: 36px;
      height: 36px;
      margin - right: 5px;
      border: 2px solid black;
      background: #ccc;
      color: #fff;
      text - align: center;
      line - height: 36px;
      font - size: 20px;
      border - radius: 20px;
    }
    /* E:nth - of - type(n)选择器 */
    .wrap dd:nth - of - type(2n) {
      background: #39f;
    }
    /* E:nth - last - of - type(n)选择器 */
    .wrap dd:nth - last - of - type(3n) {
      border: 2px solid red;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <dl class = "clearfix">
      <dd> 1 </dd>
      <dt> 2 </dt>
      <dd> 3 </dd>
      <dd> 4 </dd>
      <dt> 5 </dt>
      <dt> 6 </dt>
      <dd> 7 </dd>
      <dd> 8 </dd>
      <dt> 9 </dt>
      <dd> 10 </dd>
```



```
</dl>
</div>
</body>
</html>
```

代码解析：

在类名为 wrap 的元素中,序列为偶数的 dd 设置蓝色背景。文本内容为 1、3、4、7、8、10 的元素为 dd 元素,偶数的 dd 指的是文本内容为 3、7、10 的这三个元素,为这三个元素设置蓝色背景。为倒数序列号为 3 的倍数的元素(即 7、1)设置红色边框。

显示效果如图 13.6 所示。



图 13.6 CSS3 结构类伪类选择器选择效果 02

部分结构类伪类选择器如表 13.5 所示。

表 13.5 结构类伪类选择器 03

选择器语法	选择器含义
E:only-child	匹配父元素下仅有的一个子元素,等同于:first-child:last-child 或:nth-child(1):nth-last-child(1)
E:only-of-type	匹配父元素下使用同种标签的唯一一个子元素,等同于:first-of-type:last-of-type 或:nth-of-type(1):nth-last-of-type(1)
E:empty	匹配一个不包含任何子元素的元素,注意,文本节点也被看作子元素
E:not(s)	匹配不符合当前选择器的任何元素

2. 与用户界面相关的伪类

与用户界面相关的伪类选择器如表 13.6 所示。

表 13.6 与用户界面相关的伪类选择器

选择器语法	选择器含义
E:enabled	匹配表单中激活的元素
E:disabled	匹配表单中禁用的元素
E:checked	匹配表单中被选中的 radio(单选框)或 checkbox(复选框)元素
E:selection	匹配用户当前选中的元素

这些选择器主要针对 HTML 中的 Form 元素操作,最常见的是表单中的文本框存在“enabled”和“disabled”两种状态,前者为可写状态,后者为不可写状态;单选钮和复选框有“checked”和“unchecked”两种状态,前者为选中状态,后者为未选中状态。如果希望将“disabled”的文本框与别的文本框区别出来,就可以这样应用。

代码实例：

```
input[type = "text"]:disabled {
    border: 1px solid #999;
```

```
background-color: #fefefe;
}
```

#### 代码解析：

为页面中禁用的文本框设置了 1 像素实线边框(颜色为 #999), #fefefe 颜色的浅灰色背景。

备注：关于其他几个伪类选择器，用法与上面的案例类似，在这里就不再列举了。

### 13.3.4 CSS3 选择器的问题区

#### 1. CSS3 选择器的使用频率

由于 CSS3 选择器得到了 IE9+ 等主流浏览器的支持，在移动端能够广泛地使用。但是在 PC 端，需要考虑 IE8-浏览器的兼容问题，如果项目要求不需要兼容 IE8-浏览器，则可以放心使用。

在结构标签选用、嵌套合理时，使用到 CSS3 选择器的概率并不高，开发时可以将 CSS3 选择器作为辅助类选择器来使用，优化代码，不建议通篇使用或滥用 CSS3 选择器。

#### 2. nth-child 与 nth-of-type 的区别

:nth-child() 与 :nth-of-type() 的选择器很类似，但是并不相同。

p:nth-child(1) 选择器指的是：父级元素中的第一个子元素，同时这个子元素必须是 p 元素。如果父级元素当中第一个子元素不是 p 元素，则不能够被选到。

p:nth-of-type(1) 选择器指的是：父级元素当中所有 p 元素中的第一个元素。

#### 代码范例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - nth-child 与 nth-of-type</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box p:nth-child(1) {
      background: #39f;
    }
    .box p:nth-of-type(1) {
      border: 2px solid black;
    }
  </style>
</head>
<body>
  <div class = "box">
    <h1>h1 标题 01</h1>
    <p>段落 01</p>
    <p>段落 02</p>
    <p>段落 03</p>
```



```
<p>段落 04 </p>
<h1>h1 标题 02 </h1>
<p>段落 05 </p>
<h1>h1 标题 03 </h1>
</div>
</body>
</html>
```

显示效果如图 13.7 所示。

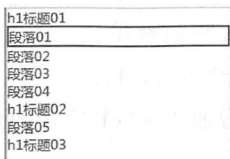


图 13.7 nth-child 与 nth-of-type



## 13.4 CSS3 圆角边框

### 13.4.1 圆角边框——border-radius

#### 1. border-radius 属性

属性功能：

为元素设置平滑拐角的显示区域。IE9+以及各个主流浏览器均支持。

代码实例：

```
border-radius: 10px;
```

代码解析：

为元素的 4 个角设置 10 像素的圆角效果。

#### 2. “圆角边框”语法与解析

基本语法：

```
border-radius: none | <length>{1,4} [/ <length>{1,4} ]?
```

基本语法中符号的含义如表 13.7 所示。

表 13.7 圆角边框语法中符号的含义

符 号	含 义
none	没有圆角(默认情况)
	表示“或”
< length >	表示长度,可以使用长度值

续表

符 号	含 义
{1, 4}	{ }里的数字代表属性值的数量范围
<length>{1, 4}	表示使用长度值,这个值可以出现 1~4 次
[ ]	中括号中的内容表示可选,也就是可以出现,也可以不出现
?	代表它前面的内容最多只能出现一次
[ ]?	表示中括号中的内容可以出现 0 或 1 次

综合分析：

默认情况下标签(或元素)并没有圆角效果；

可以为其设置 1~4 个值(<length>{1, 4})；

在 1~4 个值的基础上,可以再增加 1~4 个值,后面的值与前面的值使用“/”符号分隔开([/ <length>{1, 4}])。

3. 借助 margin 理解 border-radius

本节旨在借助 border-radius 讲解一种学习方法。该方法是通过观察效果实例并借助习得的旧知识,大胆推测属性的基本特点,从而掌握新知识,并建立新知识与旧知识之间的联系,让记忆和理解更加牢固。如果想直接查看 border-radius 的具体结论,请跳过此部分,查看下一节即可。

效果图 1 如图 13.8 所示。

效果图分析：

A 当中,4 个角的样式是相同的；B 当中,左上角和右下角的样式相同、右上角和左下角的样式相同；C 当中,右上角和左下角样式相同,拥有比较大的弧度,左上角有比较小的圆角,右下角没有圆角；D 当中,4 个方向的圆角程度都不同。

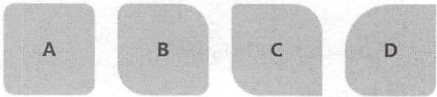


图 13.8 圆角边框效果图 1

结论推断：

4 个角可以设置为不一样的圆角,可以推测出 border-radius 有可能存在 4 个用于控制圆角的属性值。

同类知识类比：

在学过的盒模型的各类属性当中,外边距、内边距、边框,都具有 4 个方向,4 个值。

以外边距为例,外边距 margin 可以设置 1~4 个值,不同的属性值使用空格进行分隔。

当拥有 4 个值时,4 个值分别表示上、右、下、左(从上部开始,顺时针方向旋转)。

border-radius 4 个值的猜测：

由于圆角边框与 margin 等属性很类似,因此可以猜测 border-radius 也拥有 1~4 个值,每个值之间使用空格分隔。

当存在 4 个属性值时,margin 是从顶部开始顺时针方向来计算；查看一个元素时,通常都是从这个元素的左上角开始,border-radius 的 4 个方向,并不是上下左右,而是左上、右上等 4 个角。因此,可以推测的是：border-radius 在设置 4 个值时,分别表示左上角、右上角、右下角、左下角。

border-radius:40px 0 10px 20px;代码表示：左上角圆角值为 40 像素,右上角为 0 像



素,右下角为 10 像素,左下角为 20 像素。

#### 关于缩写方式的推断:

根据 margin 的缩写方式,推测 border-radius 缩写规则如下。

一个值时,表示 4 个角值相同。

两个值时,第一个值表示左上角和右下角(↖↘),第二个值表示右上角和左下角(↗↙)。

三个值时,第一个值表示左上角(↖),第二个值表示右上角和左下角(↗↙),第三个值表示右下角(↘)。

四个值时,分别表示左上角(↖)、右上角(↗)、右下角(↘)、左下角(↙)。

使用 border-radius 实现 A~D 的圆角效果。

#### 代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - CSS3 圆角边框</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap div {
      float: left;
      width: 80px;
      height: 80px;
      margin: 10px;
      background: # d5d2c1;
      line - height: 80px;
      text - align: center;
      font - weight: bold;
    }
    .box1 {
      border - radius:10px;
    }
    .box2 {
      border - radius: 10px 25px;
    }
    .box3 {
      border - radius: 5px 25px 0;
    }
    .box4 {
      border - radius: 40px 0 10px 20px;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = 'box1'>A</div>
    <div class = 'box2'>B</div>
    <div class = 'box3'>C</div>
    <div class = 'box4'>D</div>
  </div>
</body>
</html>
```

效果图 2 如图 13.9 所示。

#### 效果图分析：

在看过 E 和 F 之后，相信你会感觉到，之前的结论需要补充一些东西，因为边框圆角在水平和垂直方向上可以不同！



图 13.9 圆角边框效果图 2

#### 结论推断：

水平方向有 1~4 个值，拥有不同的缩写方法；垂直方向也有 1~4 个值，也拥有相应的缩写方法。

在不设置垂直方向的值时，默认垂直方向和水平方向相同(A~D 的效果)。

#### 最后一个未解决的问题：

如何区分水平方向和垂直方向的值？

这个问题没有办法根据查看效果进行推断。只能够通过查看语法，在 border-radius 的语法中，规定使用/实现水平和垂直方向数值的分隔。/之前为水平方向的值，/之后为垂直方向的值，每个方向均有 1~4 个值，各自遵从相应的缩写。如果不书写垂直方向的值时，默认与相应水平方向的值相同。

使用 border-radius 实现 E 和 F 的圆角效果。

#### 代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - CSS3 圆角边框水平垂直推理</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap div {
      float: left;
      margin: 10px;
      background: # d5d2c1;
      line-height: 80px;
      text-align: center;
      font-weight: bold;
    }
    .box1 {
      width: 80px;
      height: 80px;
      border-radius: 15px / 40px;
    }
    .box2 {
      width: 180px;
      height: 80px;
      border-radius: 90px / 40px;
    }
  </style>
</head>
<body>
  <div class = "wrap">
```



```

        <div class = 'box1'>E</div>
        <div class = 'box2'>F</div>
    </div>
</body>
</html>

```

**备注：**单纯的记忆结论并不难，但是却比较容易遗忘。对于接触的很多知识，可以通过思考和尝试，一方面加深对代码的理解，另一方面也会让记忆变得更持久。

#### 4. 关于圆角边框的完整“书写规则”

- (1) 在默认情况下，标签并没有圆角；
- (2) 可以为元素设置 1~8 个值，来控制元素的圆角效果；
- (3) 1~7 个值的情况时均为 8 个值的一种缩写形式；
- (4) 8 个值当中，前 4 个值代表水平方向，后 4 个值代表垂直方向，水平与垂直方向使用“/”来进行分隔，每个值之间使用空格分隔；
- (5) 如果不书写后 4 个值时，水平和垂直方向的圆角值相同；
- (6) 水平和垂直方向的 4 个值均具有缩写形式，缩写规则类似于 margin，(水平/垂直方向)一个值时，表示 4 个角；(水平/垂直方向)两个值时，第一个值表示左上角和右下角，第二个值表示右上角和左下角；(水平/垂直方向)三个值时，第一个值表示左上角，第二个值表示右上角和左下角，第三个值表示右下角；(水平/垂直方向)4 个值时，分别表示左上角、右上角、右下角、左下角(从左上角顺时针选择角)；
- (7) 水平和垂直方向，各自的缩写不互相影响，即可以水平方向设置一个值，垂直方向设置三个值。

### 13.4.2 圆角边框效果实例

**功能需求：**

实现图 13.10 中的黑色扇形效果。

**代码实例：**



图 13.10 扇形效果

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF-8">
    <title>HTML5 布局之路 - CSS3 圆角 - 扇形效果</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        .box {
            width: 0px;
            height: 0px;
            border: 50px solid transparent;
            border-top-color: black;
            border-radius: 50%; /* 此处也可以使用 50px */
        }
    </style>

```

```

</head>
<body>
    <div class="box"></div>
</body>
</html>

```

#### 代码解析:

扇形,其实就是四分之一矩形(三角形)的两个角变成圆角。所以首先要实现四分之一矩形的效果,在第3章边框中曾讲解过边框的显示效果,因此此处将元素的宽度、高度设置为0,之后设置4个方向的边框,只给顶部边框设置黑色,其他方向边框设置为透明。最后,设置圆角边框为水平、垂直方向的50%,就构成了一个扇形。

### 13.4.3 CSS3 圆角带来的变革

在CSS3圆角边框出现之前,对于网页中出现“圆状拐角”的边框,均需要通过“切图”并将图片设置为某个元素的背景图的方式来实现。以如下需求为例,通过传统方法和CSS3圆角边框的新方法分别进行实现,看看CSS3圆角边框这一属性为开发带来了什么新变化。

功能需求如图13.11所示。

#### 需求分析:

图像的宽度×高度=450×450;

图像外侧有15像素的浅灰色(色值:#f0f0f0)圆环;

浅灰色圆环外侧还有5像素的深灰色(色值:#cccccc)圆环。



图 13.11 圆环头像效果

#### 1. 传统方法实现圆角背景

##### 代码实例:

```

<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 非圆角实现圆角背景效果</title>
    <link rel="stylesheet" href="../../css/reset.css">
    <style>
        .box {
            width: 450px;
            height: 450px;
            padding: 20px;
            background: url('../images/bac.png') center center no-repeat;
        }
        .box img {
            display: block;
            width: 100%;
            height: 100%;
        }
    </style>
</head>

```



```
<body>
  <div class="box"></div>
</body>
</html>
```

#### 代码解析:

想用传统方法实现这个功能,需要有外侧的圆环图像。将两层圆环作为一张图片,使用 Photoshop 切下来,设置为父级元素的背景图。

#### 方法的劣势:

(1) 额外的文件。由于需要设置背景图,因此增加了额外的文件,文件大小增大,增加了一次 HTTP 请求,网页加载速度变缓。

(2) 图像发布(维护)极其不便。由于要保证每张图像能够在圆环当中正常显示,因此,对于图像的像素精确度要求极高,一个像素都不能有偏差。同时,还需要使用 Photoshop 针对图像进行处理,将图像周围处理为半透明,并将图像存储为 png 格式。一系列的操作增加了后台维护人员的额外工作。

(3) 不利于后期修改维护。一旦后期需求发生调整,例如,圆环颜色发生变化,此时前端工程师必须重新切图。如果是图像大小发生改变(从  $450 \times 450$  调整成  $440 \times 440$ ),那意味着所有的图像都需要重新处理、裁剪。

使用到的素材图(png 格式)如图 13.12 所示。

#### 2. CSS3 方法实现圆角背景

CSS3 提供了更简便快捷的属性 border-radius 之后,实现边框圆角就非常简单了,而且多了很多优点:减少了代码,提高了维护性,减少了图片的 HTTP 的请求,提升了网站的性能。

#### 代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 使用圆角边框实现效果</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    .box {
      width: 450px;
      height: 450px;
      padding: 15px;
      border: 5px solid #ccc;
      background: #f0f0f0;
      border-radius: 50%;
    }
    .box img {
      display: block;
```

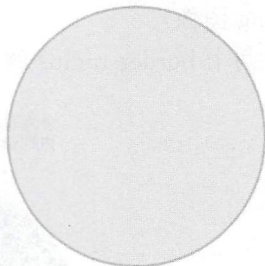


图 13.12 圆环头像效果中的素材图

```
        width: 100%;  
        height: 100%;  
        border-radius: 50%;  
    }  
    </style>  
</head>  
<body>  
    <div class = "box"><img src = "../images/demo_pic.png" alt = "测试用图" title = ""></div>  
</body>  
</html>
```

#### 代码解析:

使用背景颜色来实现浅灰色圆环部分,使用边框颜色来实现深灰色圆环部分,再使用圆角边框将原有的矩形框“变成”圆形。

此处需要注意的是,外部 div 和内部的 img 标签均需要添加 border-radius 属性,外部 div 设置圆角边框的主要目的是让深色和浅色区域均能够以圆环效果显示,img 设置圆角边框的主要目的是让图片以圆形方式显示(无论图像采用的是矩形图像还是切图处理过的圆形 png 图像)。

两个 border-radius 的作用图解如图 13.13 所示。

外部元素没有设置border-radius,  
内部元素设置border-radius: 50%;



(a)

外部元素设置border-radius: 50%,  
内部元素没有设置border-radius;



(b)

外部元素设置border-radius: 50%,  
并设置overflow: hidden;内部元素  
没有设置border-radius。



(c)

外部元素设置border-radius: 50%,  
内部元素设置border-radius: 50%。



(d)

图 13.13 圆环头像效果中内外元素设置圆角的显示效果



当为外部(父级)元素设置 border-radius 时,是对外部元素显示范围的控制;为图像设置 border-radius 时,是对图像显示范围的一种控制。

在图 13.13(a)中,对图像的显示范围进行了控制,但是并没有控制外部元素的显示范围,因此并没有产生圆环的背景效果;

在图 13.13(b)中,对外部的显示范围进行了控制,但是没有控制图像的显示范围。由于图像有可能没有进行相应裁剪(圆形显示区域、其他区域半透明的 png 图像),因此,图像的其他部分会超出;

在图 13.13(c)中,在图 13.13(b)的基础上添加了 overflow: hidden,这种处理方法对于没有内边距的父级元素是可行的,但是对于有内边距的父级元素,就是当前的显示效果;

在图 13.13(d)中,对外部元素和图像元素的显示范围均进行了控制,是想达到的最终效果。

#### 方法的优势:

(1) 使用代码代替图像。原本需要使用图像的部分,改成了两行代码,大大缩小了文件的大小,同时并不会涉及服务器的请求,因此,对于网站加载速度也会有所提升。

(2) 便于维护和修改。如果想改变圆环区域的大小,操作就变得简单了很多,只需要改变外部元素的 padding 值和 border 值即可,不需要再切图,也不需要修改其他元素的样式,方便快捷。

(3) 后台的数据发布变得简单。由于对 img 元素也进行了 border-radius 的设置,使其显示区域得到了控制,因此,后台在发布数据时,不需要再针对图像进行过多的处理(透明背景、png 格式等),只需要将图片批处理为相应尺寸即可。

### 13.4.4 CSS3 圆角边框的问题区

(1) 想要使用圆角边框是不是必须设置 border?

不需要,border-radius 与 border 没有直接的关联,容易产生误导的原因在于圆角边框中的语法出现了“border”一词。border-radius 控制除外边距的盒模型区域。

(2) 如果为图像(img 标签)外部的标签设置圆角边框,是否能够限制图像的显示?

单独对外部标签设置 border-radius,不能限制图像的显示。

如果外部标签没有内边距,只有内容区(width 和 height)与边框,此时可以为外部元素设置圆角边框,并添加 overflow: hidden;以实现图像显示区域限制的需求;如果外部标签存在内边距,则必须为图像标签设置自身的圆角边框,才能够达到期望的显示效果。



## 13.5 CSS3 文本阴影

### 13.5.1 文本阴影——text-shadow

text-shadow 属性推出之前,对于网页中的一些特殊样式的字体(字体类型为微软雅黑、宋体等常见字体,但是样式比较特殊),都需要通过图片的形式展示在网页当中。在这个过程中,必然会涉及 PS 的切图处理,会花费不少的时间,同时,后期的维护也并不方便,随意更改其中一个字,就需要再次切图,同时,如果是比较重要的一些文字,当转换为图片时,对

SEO 也是有一定的“弊端”的。

新增的文本阴影属性,让之前的一系列繁杂操作变得简单可控,图 13.14 中所展示的就是几种比较常见的,能够使用文本阴影实现的文字样式。



图 13.14 各类文本阴影特效

1. text-shadow 属性

属性功能:

为文本设置阴影,阴影并不占据任何物理空间。IE10+以及各个主流浏览器均支持。

代码实例:

```
text-shadow: 10px 10px red;
```

代码解析:

为文本设置红色阴影,阴影向右偏移 10 像素,向下偏移 10 像素。

2. “文本阴影”语法与解析

基本语法:

```
text-shadow: none | shadow [ , shadow ] *  
shadow = length{2,3} && color?  
默认值: none
```

基本语法中符号的含义:

在文本阴影的语法当中,还有一些之前在圆角边框中没有提到符号,如表 13.8 所示。

表 13.8 文本阴影语法中符号的含义

符 号	含 义
*	表示出现多次
[ , shadow ] *	表示中括号中的内容可以出现 0 次到多次
&.&	表示“与”

综合分析:

text-shadow 为文本阴影的属性。

默认情况下,text-shadow 的属性值为 none,即没有阴影;



如果为 `text-shadow` 设置阴影,可以设置两三个长度值和一个颜色值;  
第三个长度值和颜色值是可选的。

### 3. 借助生活中的阴影理解 `text-shadow`

本节旨在借助 `text-shadow`,讲解一种学习方法。该方法是通过借助生活中的事物来辅助对技术知识的理解,需要寻找具有特点相同或相近的事物进行观察,并大胆推测、通过代码实践验证自己的想法,让记忆和理解更加牢固。如果想直接查看 `text-shadow` 的具体结论,请跳过此部分,查看下一节即可。

#### (1) 生活中的阴影有什么特点?

当人不同时间站在阳光下时,影子的位置并不相同;

多个光源时,人可以存在多个影子;

影子可以是清晰可见的,也可以是模糊的轮廓;

在不同光源下,影子的颜色也并不相同;

影子并不占据任何空间。

#### (2) 推测 `text-shadow` 具有的功能。

可以设置一个或多个阴影;

每个阴影应当包含位置、模糊度和颜色;

对于位置来说,由于网页是二维平面,因此应当包括两个值,分别表示水平方向和垂直方向的阴影位置;

阴影不占据物理空间。

### 4. 关于文本阴影的完整“书写规则”

(1) 在默认情况下,文本没有阴影;

(2) 每一种文本可以设置多个阴影;

(3) 每一个阴影包含 4 个属性值,分别是水平偏移量、垂直偏移量、阴影模糊值和阴影颜色;

(4) 如果设置阴影,阴影的水平偏移量和垂直偏移量必须书写,模糊值和阴影颜色可选;

(5) 以文本的左上角作为二维坐标系的原点,水平正方向向右,垂直正方向向下;水平偏移量设置为正值时,阴影出现在文本的右侧,设置为负值时,阴影出现在文本的左侧;垂直偏移量设置为正值时,阴影出现在文本的下面,设置为负值时,阴影出现在文本的上面;

(6) 如果不书写模糊值,则默认为不模糊,模糊值需要书写单位,通常为像素(px),不允许为负值;

(7) 如果不书写阴影颜色,则阴影默认为文字颜色;

(8) 阴影不占据物理空间。

## 13.5.2 文本阴影效果实例

功能需求:

实现图 13.15 中的立体文字效果。

# HTML5布局之路

图 13.15 立体文字——文本阴影特效

代码实例：

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 文本阴影</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      width: 400px;
      height: 150px;
      background: #f0f0f0;
      text-align: center;
      line-height: 150px;
      color: #fff;
      font-weight: bold;
      font-size: 40px;
      text-shadow:
        0px 1px 0px #ccc,
        0px 2px 0px #c9c9c9,
        0px 3px 0px #bbb,
        0px 4px 0px #b9b9b9,
        0px 5px 0px #aaa,
        0px 6px 1px rgba(0, 0, 0, 0.1),
        0px 0px 5px rgba(0, 0, 0, 0.1),
        0px 1px 3px rgba(0, 0, 0, 0.3),
        0px 3px 5px rgba(0, 0, 0, 0.2),
        0px 5px 10px rgba(0, 0, 0, 0.25);
    }
  </style>
</head>
<body>
  <div class = "box">HTML5 布局之路</div>
</body>
</html>
```

代码解析：

为文本设置了 10 个阴影，前面几个阴影主要是制作成多层阴影的叠加效果，后面几个阴影主要是为文字添加一些模糊的阴影，从而增加立体感。



**备注：**背景颜色和文字颜色以及阴影颜色的配合，才能够实现出一些视觉上的效果，所以，在尝试这个代码实例时，建议先按照当前案例值设置，查看效果之后再进行调整和自我创作。

### 13.5.3 文本阴影的问题区

(1) 当文本阴影的某些属性值为0时，px能否省略？

可以。如下三行代码等价：

```
text-shadow: 0px 1px 0px #ccc;
text-shadow: 0 1px 0 #ccc;
text-shadow: 0 1px #ccc;
/* 由于模糊值默认为 0px, 所以可以省略 */
```

(2) 当一个文本存在多个阴影时，代码能否换行？

可以，但是需要注意符号不要使用错误。CSS 代码每一句属性值的设置，均是以分号(;)作为结束的，如果换行，则不要在换行位置书写分号。

对于一个文本存在多个阴影的情况，推荐将代码换行，并且调整好缩进（具体缩进的Tab次数并没有规定，但是至少一次），这样更有助于代码的阅读（如同上面立体文字效果实例的代码书写方式）。

(3) 文本阴影的设置似乎对设计和美学有一定的要求，不是太了解怎么办？

在文本阴影的设计与实现当中，前端开发工程师只负责实现，而由设计师进行样式的设计和调整，当在文本阴影值设置中遇到“效果和设计图不太一致”又不知道如何调整时，就去和设计师沟通一下，请她（他）帮个忙吧！

337



## 13.6 CSS3 盒阴影

### 13.6.1 盒阴影——box-shadow

#### 1. box-shadow 属性

盒阴影和文本阴影类似，所不同的是，盒阴影指的是元素的阴影，在属性方面，比文本阴影多增加了两种属性（阴影外延值和阴影位置）。

**属性功能：**

为文本设置阴影，阴影并不占据任何物理空间。IE9+以及各个主流浏览器均支持。

**代码实例：**

```
box-shadow: 10px 0 black;
```

**代码解析：**

设置水平向右偏移 10 像素的黑色盒阴影。

## 2. “盒阴影”语法与解析

基本语法：

```
box-shadow: none | shadow [ , shadow ] *  
shadow = length{2,4} && color? && pos?  
默认值: none
```

综合分析：

box-shadow 为盒阴影的属性。

默认情况下,box-shadow 的属性值为 none,即没有阴影；

如果为 box-shadow 设置阴影,可以设置 2~4 个长度值、一个颜色值、阴影位置；  
第 3、4 个长度值、颜色值、位置是可选的。

### 3. 关于盒阴影的完整“书写规则”

(1) 在默认情况下,标签(或元素)没有盒阴影；

(2) 每一个标签可以设置多个阴影；

(3) 每一个阴影包含 6 个属性值,分别是水平偏移量、垂直偏移量、阴影模糊值、阴影外延值、阴影颜色和阴影位置,不同属性值之间使用空格分隔；

(4) 属性的书写顺序可以是：“阴影位置 水平偏移量 垂直偏移量 阴影模糊值 阴影外延值 阴影颜色”；也可以是“水平偏移量 垂直偏移量 阴影模糊值 阴影外延值 阴影颜色 阴影位置”；

(5) 如果设置阴影,阴影的水平偏移量和垂直偏移量必须书写,模糊值、外延值、阴影颜色、阴影位置可选；

(6) 以文本的左上角作为二维坐标系的原点,水平正方向向右,垂直正方向向下；水平偏移量设置为正值时,阴影出现在标签的右侧,设置为负值时,阴影出现在标签的左侧；垂直偏移量设置为正值时,阴影出现在标签的下面,设置为负值时,阴影出现在标签的上面；

(7) 如果不书写模糊值,则默认为不模糊,模糊值需要书写单位,通常为像素(px),不允许为负值；

(8) 如果不书写外延值,则默认为 0,外延值指的是在当前基础上,4 个方向均扩展的大小,需要书写单位,通常为像素(px),不允许为负值；

(9) 如果不书写阴影颜色,则默认为文字颜色；

(10) 如果不书写阴影位置,则默认为外阴影,如果想设置内阴影,需使用 inset；

(11) 阴影不占据物理空间。

## 13.6.2 盒阴影的效果实例

### 1. 基本案例

功能需求：

实现图 13.16 中的盒状阴影显示效果。



## HTML5布局之路

图 13.16 盒阴影特效

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 盒阴影特效</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      width: 400px;
      height: 150px;
      background: #f0f0f0;
      text-align: center;
      line-height: 150px;
      color: #39f;
      font-weight: bold;
      font-size: 40px;
      box-shadow: 0px 0px 10px 5px #39f inset;
    }
  </style>
</head>
<body>
  <div class = "box">HTML5 布局之路</div>
</body>
</html>
```

代码解析：

阴影处于标签内部，因此阴影位置应当设置为 inset；4 个方向上的阴影值相同，因此并不需要设置水平偏移量和垂直偏移量，只需要设置阴影外延值。由于在效果中，阴影并不清晰，因此需要设置阴影模糊值。

### 2. 使用盒阴影优化圆角边框中的实例

有了盒阴影，之前的这种效果会不会变得更简单呢？圆角边框时使用的圆环头像特效如图 13.17 所示。



图 13.17 圆角边框时使用的圆环头像特效

## 代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 盒阴影实现圆环</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .pic {
      display: block;
      width: 450px;
      height: 450px;
      margin: 20px;
      box - shadow: 0px 0px 0px 15px #f0f0f0, 0px 0px 0px 20px #ccc;
      border - radius: 50 % ;
    }
  </style>
</head>
<body>
  <img class = "pic" src = "../images/demo_pic2.jpg" alt = "测试用图" title = "">
</body>
</html>
```

## 代码解析：

两个圆环均设置为元素的阴影,为元素设置圆角边框,就能够实现想要的效果。

和之前的方法相比,该方法在结构方面可以更加精简(在具体开发当中结构的精简程度与网站功能需求也有一定的关系,此处只是说明该方法并不需要使用其他标签辅助实现)。

## 3. 盒阴影有可能会被背景所遮挡

阴影并不占据空间,因此,一个元素的阴影有可能与自身、其他元素的位置发生重叠。

## 代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 盒阴影有可能被背景所覆盖</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .self div {
      width: 200px;
      height: 100px;
      margin: 10px;
      border: 2px solid black;
      box - shadow: 0px 0px 0px 10px #39f inset;
    }
    .self .box2 {
```



```

        background: #ccc;
        box-shadow: 0px 0px 0px 10px #39f inset;
    }
    .sibling {
        width: 408px;
        padding: 10px;
    }
    .sibling div {
        float: left;
        width: 200px;
        height: 100px;
        border: 2px solid black;
    }
    .sibling .box1 {
        box-shadow: 0px 0px 0px 10px #39f;
    }
    .sibling .box2 {
        background: #ccc;
    }
    .parent {
        overflow: hidden;
        width: 200px;
        margin: 10px;
        border: 2px solid black;
    }
    .parent div {
        width: 200px;
        height: 100px;
        box-shadow: 0px 0px 0px 10px #39f;
    }
}
</style>
</head>
<body>
    <div class="self">
        <div class="box1">内阴影,无背景色</div>
        <div class="box2">内阴影,设置了背景色</div>
    </div>
    <div class="sibling clearfix">
        <div>没有设置背景颜色的兄弟级元素</div>
        <div class="box1">该元素设置了外阴影</div>
        <div>没有设置背景颜色的兄弟级元素</div>
        <div class="box2">设置背景颜色的兄弟级元素</div>
        <div class="box1">该元素设置了外阴影</div>
        <div class="box2">设置背景颜色的兄弟级元素</div>
    </div>
    <div class="parent">
        <div>子元素设置了外阴影,父元素设置 overflow: hidden;</div>
    </div>
</body>
</html>

```

显示效果如图 13.18 所示。

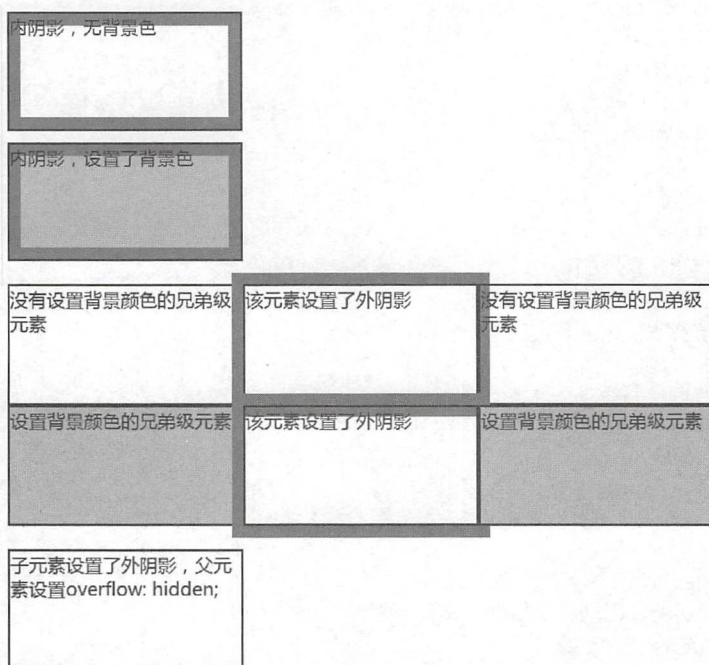


图 13.18 盒阴影可能被背景覆盖

代码解析：

如果为元素设置内阴影，阴影会覆盖在自身元素的背景之上，内容之下；

如果为元素设置外阴影，与兄弟级元素产生“重合”，如果兄弟级元素设置了背景颜色，阴影有可能会被遮盖在背景的后面，阴影是否被遮盖取决于元素与兄弟级元素的位置关系，如果设置了背景颜色的兄弟级元素在设置外阴影元素的前面，外阴影不会被覆盖，如果兄弟级元素在设置外阴影元素的后面，外阴影会被覆盖；

如果子元素设置了外阴影，其阴影超出了父元素的显示范围，而父元素设置了 `overflow: hidden`，则超出的阴影部分会被隐藏。

### 13.6.3 关于盒阴影的问题区

不是说元素并不会遮挡自身内阴影吗，为何 `img` 标签内阴影被遮挡？

`img` 标签的图片属于元素的内容区域，而非背景区域。此前的结论是：自身元素的背景并不会遮挡自身元素的内阴影，但是内容会覆盖在内阴影的上面。



## 13.7 CSS3 背景新属性

### 13.7.1 背景尺寸——`background-size`

属性功能：

设置背景图片的尺寸大小。IE9+以及各个主流浏览器均支持。



基本语法：

```
background-size: 100% 80%;
```

代码解析：

background-size 需要两个值，它的类型可以是像素(px)、百分比(%)或是 auto，还可以是 cover 和 contain。第一个值为背景图的 width，另外一个值用于指定背景图上的 height，如果只设定一个值，则第二个默认为 auto，但当值为 cover 和 contain 时除外。

此处代码表示：设置背景图的水平方向尺寸为容器宽度的 100%，垂直方向尺寸为容器高度的 80%。

属性值如表 13.9 所示。

表 13.9 背景尺寸的属性值

值	描 述
length	具体的长度值
percentage	指定百分比
cover	等比缩放到完全覆盖容器，背景图像有可能超出容器
contain	将背景图像等比缩放到宽度或高度与容器的宽度或高度相等，背景图像始终被包含在容器内
auto	真实大小

**备注：**如果使用 cover，则会保持图像的宽高比例，将图片缩放到正好完全覆盖定义的背景区域，其中有一边和背景相同，这种方法要求切图时充分考虑容器的大小，从而达到最佳的显示效果。

当指定为百分比时，大小会由所在区域的宽度、高度，以及 background-origin 的位置决定。图片大小不是按背景图片大小的百分数来计算的，而是按照装载背景图的元素的百分比来计算。

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 背景尺寸</title>
  <link rel="stylesheet" href="../css/reset.css">
  <style>
    .box {
      float: left;
      width: 200px;
      height: 150px;
      margin: 5px;
      border: 2px solid black;
      background-image: url('../images/common.jpg');
      background-repeat: no-repeat;
    }
  </style>
</head>
</html>
```

```
.cover {
    background-size: cover;
}
.contain {
    background-size: contain;
}
.perx {
    background-size: 100%;
}
.perxy {
    background-size: 100% 100%;
}
</style>
</head>
<body>
    <div class="box">默认状态</div>
    <div class="box cover">cover 状态</div>
    <div class="box contain">contain 状态</div>
    <div class="box perx">一个百分比值</div>
    <div class="box perxy">两个百分比值</div>
</body>
</html>
```

显示效果如图 13.19 所示。



图 13.19 背景尺寸不同属性值的显示效果

### 13.7.2 背景切割——background-clip

属性功能：

将背景图片做适当的裁剪，以适应需要。根据设置的盒子部位，那么图片在这个部位的外边缘以外的部分都会不可见（注意：这里并不是真正意义上的裁剪）。IE9+以及各个主流浏览器均支持。

基本语法：

```
background-clip: content-box;
```

代码解析：

进行背景切割，只保留内容区（width 和 height 区域）的部分。

属性值如表 13.10 所示。



表 13.10 背景切割的属性值

值	描 述
content-box	表示从内容区开始(width 和 height 区域)
padding-box	表示从内边距区开始(padding 区域)
border-box	表示从边框区开始(border 区域)

显示效果如图 13.20 所示。

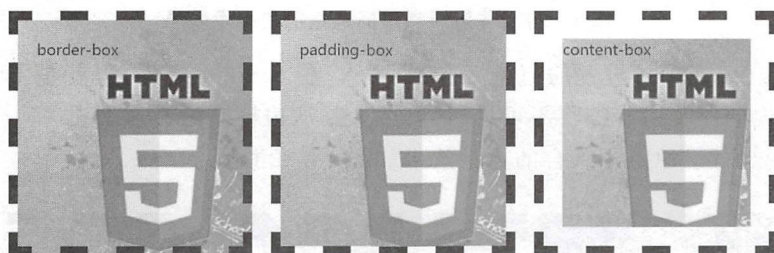


图 13.20 背景切割不同属性值的显示效果

备注：为了便于理解，可以把内边距、边框、外边距看成一个环形的容器，分别对应 border-box、padding-box、content-box。

### 13.7.3 背景原点——background-origin

属性功能：

用来决定图片的原始起始位置。IE9+以及各个主流浏览器均支持。

基本语法：

```
background-origin: content-box;
```

代码解析：

设置背景从内容区(width 和 height 区域)的左上角开始显示。

属性值如表 13.11 所示。

表 13.11 背景原点的属性值

值	描 述
content-box	表示从内容区开始，即背景图片的左上角和内容边缘左上角对齐
padding-box	默认值。表示从内边距区开始显示，即背景图片的左上角和内边距的左上角对齐
border-box	表示从边框区开始，即背景图片的左上角与边框左上角对齐

显示效果如图 13.21 所示。

### 13.7.4 背景切割与背景原点的区别

background-clip 与 background-origin 并不相同。background-origin 表示的是背景图从哪个位置开始显示，background-clip 则表示在哪个区域显示背景图。在实际开发当中，可以将两者相结合，来实现一些特殊功能或效果。例如，背景图片起始位置是从 border-box



图 13.21 背景原点不同属性值的显示效果

开始,但 background-clip 设置的值是 content-box,在 content 之外,也就是 border-box 内, padding-box 内的图片内容都将不可见,尽管图片是从边框开始显示的。

背景切割与背景原点相结合的显示效果如图 13.22 所示。

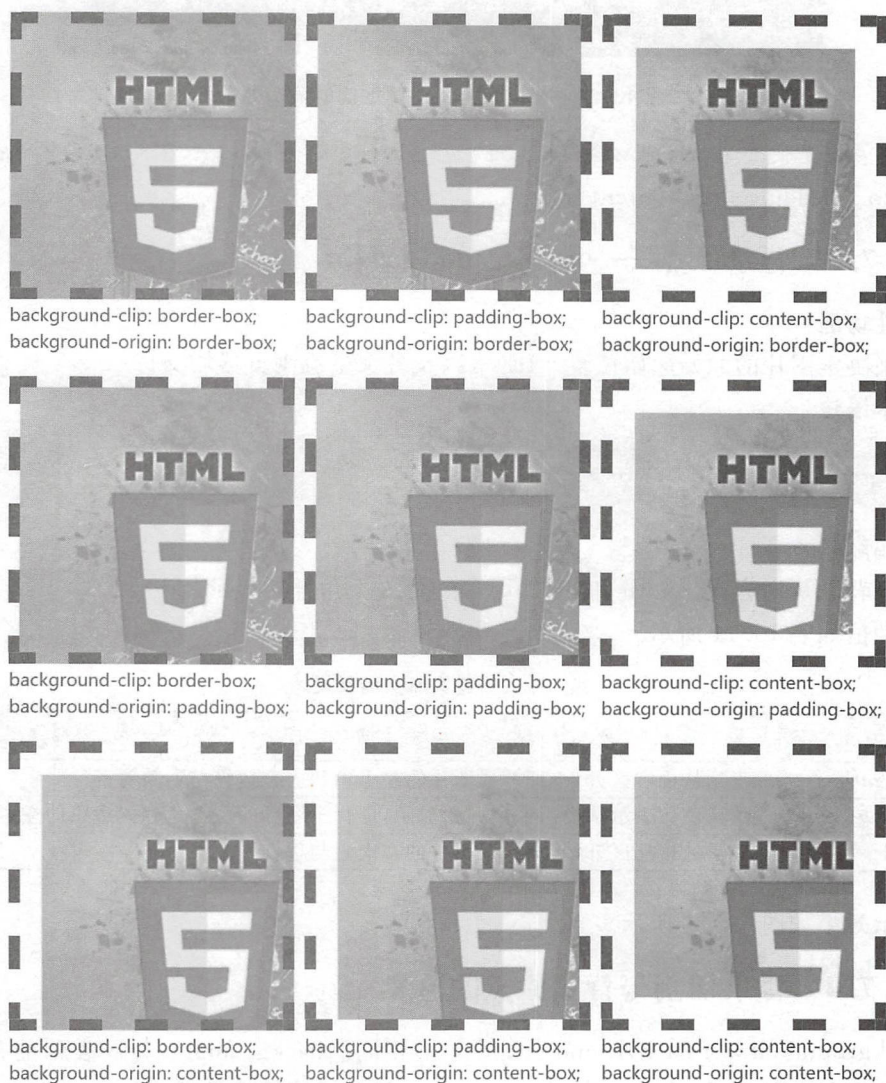


图 13.22 背景原点与背景切割结合后的显示效果





## 13.8 渐变背景

### 13.8.1 什么是渐变

渐变,可以理解为由一个颜色变化到另一个颜色。当然,也可以是多个颜色之间的变化。

在渐变属性的属性值中应当有起始变化位置、起始颜色、渐变的角度、变化到的百分比位置以及变化到的颜色。

### 13.8.2 渐变的种类

CSS3 能够实现线性渐变和径向渐变两种,如图 13.23 所示。

### 13.8.3 如何书写 CSS3 渐变

线性渐变: linear-gradient

径向渐变 radial-gradient

#### 1. 借助在线生成工具

由于各个浏览器对于渐变的书写均不相同,因此,对于渐变的处理,通常从网络中搜索 CSS3 渐变生成工具,直接生成兼容各个平台的代码。

此外,前端开发工程师在还原设计图中的渐变时,需要通过渐变面板进行查看,网络中的“渐变生成工具”提供了同样的面板,这样的操作能够让前端工程师更好地还原设计图。

CSS3 在线生成工具网址: <http://www.colorzilla.com/gradient-editor/>。

#### 2. 在线渐变生成工具的用法

在 Photoshop 设计图中,设计师会通过这样两种方式制作渐变,分别是“渐变填充”方式创建渐变效果、以图层样式方式添加渐变效果,在 Photoshop 软件中能够寻找到这两类“图层”,如图 13.24 所示。

注: Photoshop 的基本使用方法在第 1 章当中已经进行了详细讲解。

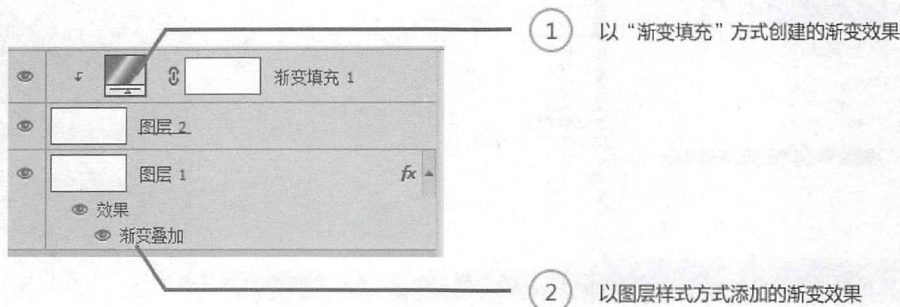


图 13.24 Photoshop 中的两种不同方式添加的渐变

方法 1, 双击“图层缩览图”, 会弹出“渐变填充”对话框, 在该对话框中, 会显示渐变的类型以及角度(从上向下, 还是从左到右等)的信息。之后, 单击“渐变填充”对话框中的“渐变”, 将调取出“渐变编辑器”, 如图 13.25 所示。

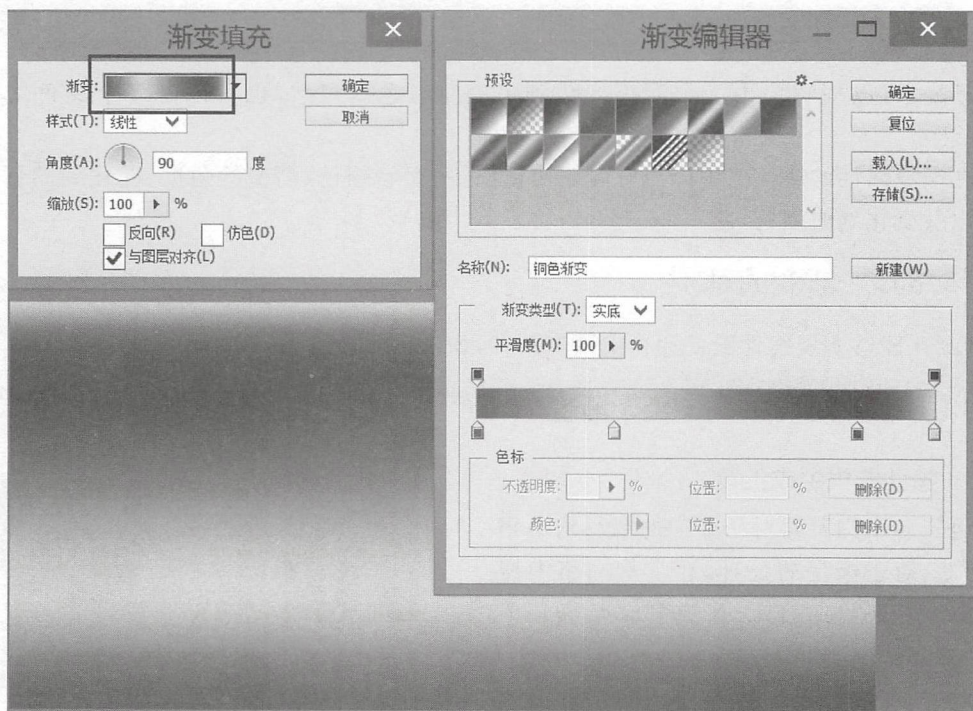


图 13.25 调取出 Photoshop 中的渐变对话框方法 1

方法 2, 双击“渐变叠加”部分, 将调取出“图层样式”对话框, 在该对话框中, 会显示渐变的类型以及角度(从上向下, 还是从左到右等)的信息。之后, 单击渐变后的颜色部分, 将调取出渐变编辑器, 如图 13.26 所示。

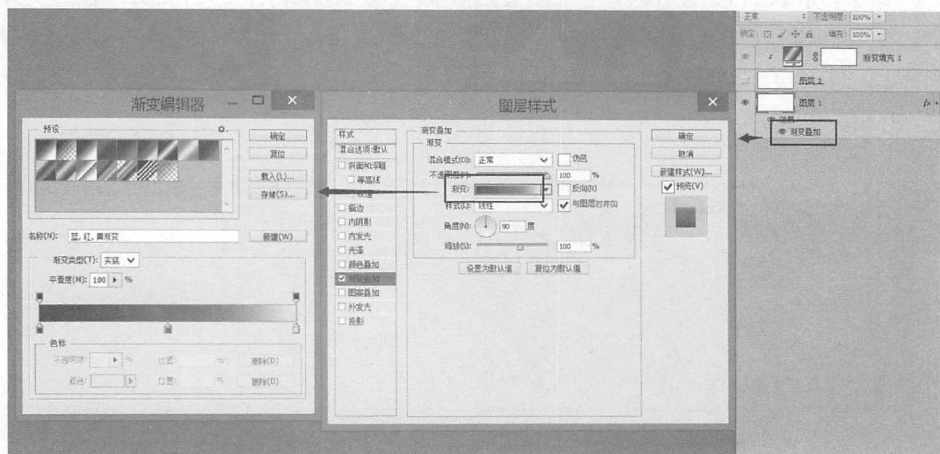
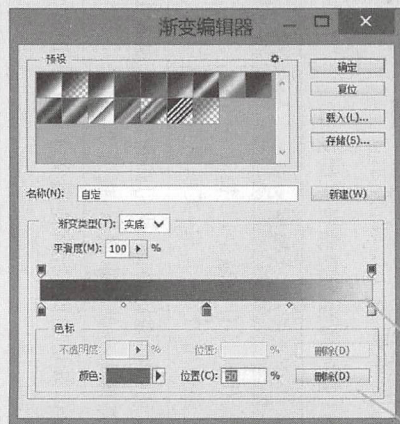


图 13.26 调取出 Photoshop 中的渐变对话框方法 2

在线生成工具与 Photoshop 渐变面板对比如图 13.27 所示。



Photoshop软件中的渐变编辑器



- ④ 如上两种渐变控制器操作方法相同  
游标尺上方的色标, 用于控制透明度  
游标尺下方的色标, 用于控制颜色  
上方的色标控制点包含透明度和位置两种属性  
下方的色标控制点包含颜色和位置两种属性  
色标的具体属性值设置在下面的色标区域

通过单击游标尺上下的色标, 来进行选择  
在游标尺上方或下方, 可以通过单击增加色标  
选中色标后, 可以进行拖曳, 移动位置  
选中色标后, 可以将其拖曳出游标尺, 实现删除

CSS3在线生成工具界面

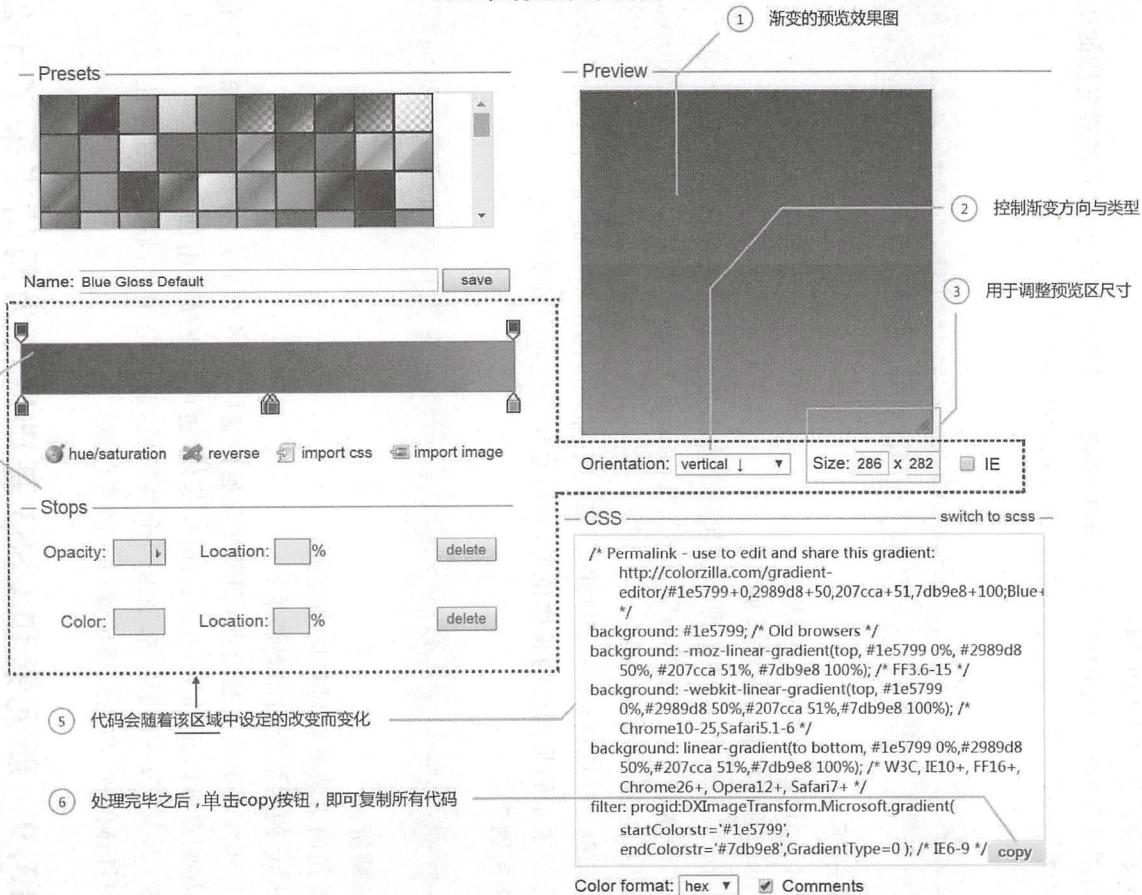


图 13.27 在线渐变生成器与 Photoshop 的渐变对话框对比图

关于渐变具体语法的解释：

不同浏览器对于渐变的书写方法并不相同，在此选用当前主流浏览器支持的语法，介绍渐变的基本语法，以保证在阅读代码时能够理解代码含义。

代码实例 1：

```
background: -webkit-linear-gradient(left top, #5dbf8c 0%, #8ecb84 25%, #bad97a 50%, #dab470 75%, #f08f67 100%);
```

代码解析：

该段代码语法是针对线性渐变的：

-webkit 的内核前缀需要添加；

left top 表示的是起始点，即渐变起始于左上角（第一个值表示水平方向，第二个值表示垂直方向，起始点可以使用“left”、“top”、“bottom”、“right”、“left top”、“left bottom”、“right top”、“right bottom”等）；

#5dbf8c 0%直到#f08f67 100%，每一个都是由两个值组成，前者是颜色，后者是百分比。百分比表示的是渐变的具体位置，颜色是该位置对应的颜色值。

代码实例 2：

```
background: -webkit-radial-gradient(50% 50%, #f00 20%, #ee0 90%, #ff0 95%);
```

代码解析：

该段代码语法是针对径向渐变的：

-webkit 的内核前缀需要添加；

50% 50%表示的是起始点，即渐变起始位置（第一个值表示水平方向，第二个值表示垂直方向，起始点可以使用 left、top、center 等值，也可以使用百分比）；

#f00 20%，每一组数字都是由两个值组成，前者是颜色，后者是百分比。百分比表示的渐变的具体位置，颜色是该位置对应的颜色值。



### 13.9 新元素和 CSS3 基础属性为网站带来了什么

新元素和 CSS3 基础属性为网站带来的变革如表 13.12 所示。

表 13.12 新元素和 CSS3 基础属性为网站带来的变革

新增功能	为网站带来的变革
HTML5 新元素	让结构可读性更强、代码量有所减少，并提升 SEO。 可以适当地取代过多使用的 div 元素，实现网页布局
CSS3 圆角边框	结束了遇到圆角就切图的尴尬局面，使用圆角边框实现各类弧状拐角，减少背景图的数量，降低服务器请求次数，提升维护性
CSS3 文本阴影	结束了遇到艺术字体就切图的局面，使用文本阴影实现常见的一些特效字体



续表

新增功能	为网站带来的变革
CSS3 盒阴影	由于其不占空间的特性,当网页使用百分比布局时,用其取代边框,能够让布局变得更简单(第 14 章中会再有所提及)。除此之外,还能够实现一些常见效果
CSS3 背景尺寸	让背景图的大小可以得到控制,操作性更强,在移动端得到广泛的应用(第 14 章中再具体提及)
CSS3 背景切割和背景原点	提升背景图的控制性,两者结合可以制作出满足特殊要求的一些效果
CSS3 渐变背景	取代原有切图实现渐变背景的情况

## 第14章

# 转战移动端(上)——百分比与rem



## 14.1 移动端发展

### 14.1.1 智能手机热潮

2007年1月,苹果公司推出了第一款 iPhone,从此打开了智能手机的大门。

虽然而今智能手机已经得到了人们的广泛使用,但是,最早苹果推出 iPhone 手机时,却得到了当时手机王者——诺基亚的质疑。诺基亚认为,虽然在表现上,大屏会更具备优势,但是却失去了触觉感受,用户在使用手机时并不能够得到按键的感受。

对于苹果这款划时代的产品,大部分的公司都处于观望状态,直至 2008 年 9 月,Android 推出了手机操作系统 Android 1.0 版本,智能手机才渐渐地普及开来。

在智能手机中,网页是能够以传统 PC 端样式进行展示的,而此前的网页却是另一个模样,使用过非智能手机的人,应该会对图 14.1 比较熟悉。



图 14.1 非智能手机上的 QQ 空间

在最初智能手机推出时,并没有什么企业能够非常肯定地说“智能手机能够广泛流传”。因此,并没有专门针对移动端,开发一套新的网站,只是沿用了原来 PC 端的页面。

如果直接沿用 PC 端的页面,显示效果如图 14.2 所示,但是,移动端浏览器厂商希望网页尽可能完整地显示在移动端设备当中,于是为移动端设备设置了一个“虚拟”的视口,使得设备能够尽可能完整地显示网页,因此实际的显示效果如图 14.3 所示。

直接沿用 PC 端的页面时,移动端浏览器仅能够显示网页的一部分内容。

移动端浏览器为了能够尽可能完整地显示网页,设置了一个“视口”,尽可能放大视口,以便在屏幕中显示完整的网页。

### 14.1.2 针对移动端的探索

随着移动端的快速向前发展,到 2012 年左右,人们发现并不能够将传统网页直接应用于移动端,主要涉及用户的交互特点、网页的加载速度、视觉体验效果等原因。

#### 1. 网站加载速度

在网页当中,加载速度就是王道。据统计:网页加载超过 4s,25% 的人会放弃;手机网





几年前,移动端网络的速度通常没有计算机的网速快。如果在移动端直接使用 PC 端的网页,网页代码中会存在很多不必要的部分,增加了代码的加载量,加载速度会受到很大的影响。

## 2. 用户交互特点

从交互层面来说,移动端并不会使用鼠标进行单击,而是手指,手指的区域大小远大于鼠标大小,因此“链接的响应区域”的不同导致传统网页在移动端使用起来异常的不舒服。

对于传统网页中的 hover 等事件,在移动端是派不上用场的。而在移动端,新增的一系列触摸手势和动作,在传统的网页中也是不可能提供的。

## 3. 视觉体验效果

由于最初浏览器厂商的考虑,在移动端,会将网页以不同的视口状态显示(在 14.3 节当中会详细介绍视口),将原本 320px 宽度的设备,虚拟成 980px 的网页内容大小,让 980px 宽度的网页能够正好容纳在设备当中,也可以理解为网页被等比例缩小了。此时,查看网页的信息就变得异常不方便。文字、图片过小,需要使用手指放大页面,再“拖曳”页面进行查看。

## 4. 移动端独立发展

由于这一系列的不便和问题,移动端的开发逐渐地脱离了传统 PC 端网页,开始了独立发展。

在最初移动端刚刚出现的几年期间,屏幕分辨率是 320px×480px 的天下。因此在那个时期,制作移动端网页时,首先由设计人员制作一套 320px 宽度的移动版 PSD 图,之后由前端工程师,使用像素这种固定单位实现页面的开发。

### 14.1.3 分辨率初变

在 2012 年,随着安卓机型的普及,移动端的设备分辨率开始出现变化,除了 320×480 的“标准大小”,宽度方面,更多的设备以 320px 宽度为基础不断扩大,高度方面,很少一部分设备高度小于 480px,更多的是在 480px 基础上有所扩大。

由于网页是自上而下显示的,内容自上而下排列布局,对于大部分网页来说,高度上的分辨率变化相对影响较小(但不代表没有影响);而宽度分辨率的变化,带来了较大的问题。

虽然面对大量的问题,但幸运的是:移动端分辨率刚开始发生变化时,还是有规律可循的。大部分都是横向 320px、480px、640px 和 768px。这种分辨率的规律性变化,将“响应式布局”推向了行业的“巅峰”,在 2013 年年初,响应式布局得到了大家的热切关注。

此时的前端开发工程师,可以通过 CSS 的相对度量单位(em、百分比等)实现页面横向的逐渐变化,使用响应式实现 4 种不同分辨率的典型变化;在纵向上用相对度量单位以及 JS 进行辅助。

### 14.1.4 多分辨率的处理

虽然响应式布局在一定程度上解决了多分辨率的问题,但是 2013 年之后,分辨率的发展让响应式布局“无从下手”。分辨率从原有的规律性变化,变得“无章可循”。各式各样的分辨率伴随着各类手机,如雨后春笋般冒了出来。此时,单纯的响应式布局没有办法解决这个问题,开发者开始考虑使用相对度量单位(百分比、em)来解决开发问题。

在发展的过程中,人们尝试了“横向百分比、纵向百分比”、“横向百分比、纵向 em”、“纵向 em、纵向 em”等各种方式,应该说各种方法都有其优劣势,纵向百分比存在一定的问题,em 的控制也让人觉得有些麻烦。

CSS3 中推出的 rem 这种相对度量单位,进一步优化了 em。于是人们又开始尝试使用“横向 rem、纵向 rem”、“横向百分比、纵向 rem”的代码编写方法。最终,发现“横向百分比、



纵向 rem”的方法是当前相对最好的一种开发方法。

### 14.1.5 移动端的未来

几年的发展,让移动端开发逐渐“成熟”起来,各个开发者在实现的基础上开始思考如何提升代码的开发效率,如何更好地编辑前端代码。例如,使用 rem 进行开发时,如何兼容高清屏幕?如何使用像素这种绝对度量单位,开发出适配多款移动设备的网站?

各个大型公司也开始为移动端打造专门的框架。关于其中两款比较优秀的框架,在第 15 章当中会进行讲解。随着人们对移动端的深入研究,相信移动端的开发会变得越来越便捷。

### 14.1.6 移动端发展总结 & 开发移动端的基本流程

移动端发展总结 & 开发移动端的基本流程如图 14.4 所示。

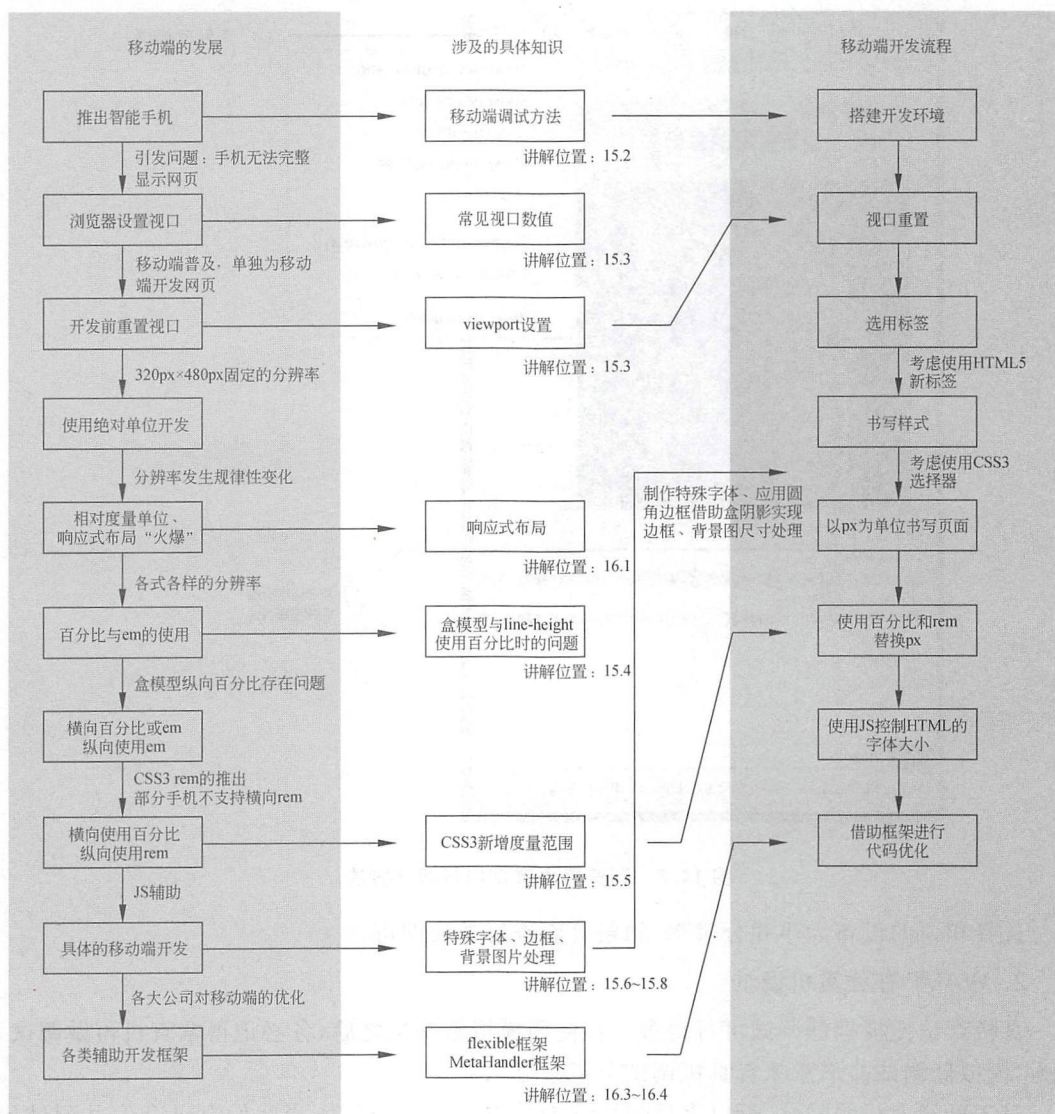


图 14.4 移动端发展总结 & 开发移动端的基本流程



## 14.2 设备调试方法

### 14.2.1 设备调试方法的种类

#### 1. 浏览器内置模拟器调试

使用谷歌等浏览器,按下 F12 键,之后单击手机的小图标(小图标位于控制台左上角),如图 14.5 所示。



图 14.5 浏览器内置模拟器调试方法

选择相应的设备大小和分辨率,刷新页面查看效果即可。

#### 2. WAMP 在线真机调试

真机调试比模拟器调试可行得多。在使用模拟器调试之后,务必记得拿真机再做调试,因为,模拟器调试并不意味着真机调试会正常显示。

对于已经上线的网站,可以直接使用移动端设备进行访问。当我们自己开发并测试代码,没有上传到服务器时,可以借助自己机器上的模拟服务器来实现(WAMP 软件)。



WAMP 模拟服务器下,真机调试的具体步骤如下。

- (1) 将代码放置在 WAMP 的 www 根目录下,打开 WAMP 实现模拟服务器端的运行;
- (2) 将 WAMP 调整为在线模式;
- (3) 将手机调整为和 PC 相同的局域网;
- (4) 在开始菜单中,执行“运行”,输入“cmd”;
- (5) 在弹出的框中输入“ipconfig”,查看当前网络的 IP 地址;
- (6) 将 IP 地址输入到手机 URL 地址栏中,实现访问。

### 14.2.2 调试的基本原则:多台真机测试

调试的基本原则:可以使用浏览器进行测试,但是一定要使用各种真机进行调试。除了在渲染方面,浏览器的模拟效果和真机上的效果有所不同之外,还需要考虑真机顶部和底部的菜单栏。



## 14.3 视口——viewport

357

### 14.3.1 视口以及常见数值

#### 1. 视口

视口中的像素指 CSS 像素,视口大小决定页面布局的可用宽度。

移动设备如果同样以浏览器(即设备屏幕)窗口作为视口,那么当前的页面布局显示就会出问题。按坐标系统显示原理,设备浏览器也在设备坐标系统规范之列,但在实际使用中,网站内容显示并没符合坐标系统规范。

将一个 980px 宽度的网页在移动端显示,预期的显示效果是:页面左侧 320px 的内容显示在设备可视区域中,同时设备可视区域出现横向滚动条,可以向右拖动查看网页的其他部分。然而,实际的显示效果却是——整个页面均显示在了设备当中(可参见之前的图 14.2 和图 14.3 中的两种显示状态)。

导致这种现象的原因在于:各个移动端设备通过 viewport(视口)把自己冒充成更宽的屏幕,大多数移动浏览器默认把布局视口的宽度设为 980px,然后,尽可能放大可见视口(布局视口宽度保持不变),以便在屏幕中显示完整的网站。

#### 2. 常见的视口数值

iPad 与 winPhone 的视口数值为 1024; iPhone 的视口数值为 980;大部分的安卓机型的视口数值为 980(由于安卓机型较多,视口数值也并非全是 980)。

### 14.3.2 调整视口大小——命令类

虽然很多移动端的设备是 320px 的,但是依旧可以将设备的视口设置为 980px,从而实现更好的显示效果。大多数移动浏览器将 HTML 页面放大为宽的视图(viewport)以符合屏幕分辨率,此处使用视图的 meta 标签来进行重置。



视口设置的两种方法：

(1) `<meta name="viewport" content="width=320, initial-scale=0.5" />`

(2) `@viewport { width: 320px; zoom: 0.5; }` / \* 该方法 Opera 和 IE 支持,需要添加内核前缀,开发中并不使用,仅作参考 \*/

### 14.3.3 viewport 元标签以及属性

#### 1. viewport 元标签及其属性

width	设定布局视口宽度
height	设定布局视口高度
initial-scale	设定页面初始缩放比例(0~10.0)
user-scalable	设定用户是否可以缩放(yes/no)
minimum-scale	设定最小缩小比例(0~10.0)
maximum-scale	设定最大放大比例(0~10.0)
target-densitydpi *	设定目标屏幕的 dpi(device-dpi)

#### 2. @viewport 规则的属性

width	设定布局视口宽度(min/max-width)
height	设定布局视口高度(min/max-height)
zoom	设定页面初始缩放比例(auto、0~10.0、%)
user-zoom	设定用户是否可以缩放(zoom/fixed)
min-zoom	设定最大放大比例(auto、0~10.0、%)
max-zoom	设定最小缩小比例(auto、0~10.0、%)
orientation	设定页面显示方向(auto、portrait、landscape)

### 14.3.4 视口调整的各种命令

设定视口宽度(固定像素值)：

```
<meta name="viewport" content="width=320" />
@-o-viewport { width:320px; }
```

设定视口宽度(可见视口宽度)：

```
<meta name="viewport" content="width=device-width" />
@-o-viewport { width:device-width; }
```

设定初始缩放比例：

```
<meta name="viewport" content="width=device-width, initial-scale=2.0" />
@-o-viewport { width:device-width; zoom:2.0; }
```





设定缩放的范围：

```
<meta name="viewport" content="width=device-width, maximum-scale=2.0, minimum-scale=0.5" />
@-o-viewport { width:device-width; max-zoom:2.0; min-zoom:0.5; }
```

禁止用户缩放：

```
<meta name="viewport" content="width=device-width, user-scalable=no" />
@-o-viewport { width:device-width; user-zoom:fixed; }
```

### 14.3.5 对待视口的基本原则

尽量禁止在移动端上的浏览器按默认 viewport 响应。在默认情况下,用手指拖曳页面时,页面会跟随手指的位置发生偏移,造成不良的用户体验。因此在开发过程中,为保证用户体验,应尽量保持分辨率和媒体设备视口大小 1:1 的比例,从而保证最佳的显示效果。

较为常用的一种视口设置：

```
<meta name="viewport" content="width=device-width, user-scalable=no">
```

359



## 14.4 当盒模型与行高遇到百分比

### 14.4.1 盒模型单位如何选择

当前,前端(或 HTML5)开发当中,主要有桌端(PC端)和移动端两个平台,桌端开发的就台式计算机、笔记本使用的网页,而移动端就是手机使用的网站与页面。

在桌端的网页开发中,像素(px)单位使用居多,如果涉及响应式、自适应等布局时会考虑使用百分比这种相对度量单位。在移动端的网页开发中,百分比、em、rem 这种相对度量单位使用居多,而像素(px)这种绝对度量单位使用较少。

### 14.4.2 margin 和 padding 使用百分比作为单位

#### 1. 特殊的 padding 与 margin

盒模型是由 width、height、border、padding、margin 这几个属性共同组成的。一个元素在网页文档中占据的真正空间是盒模型的大小,而非简简单单 width 和 height 的值。在使用百分比进行移动端开发时,盒模型存在一定的问题。

padding 和 margin 在设置百分比的时候,纵向并不是针对父级的 height 进行计算,而是参照以最近的块级祖先元素的宽度进行计算。究其原因,在于需要构建在纵横两个方向上相同的 margin/padding,如果两个百分比的相对方式不同,那用百分比就无法得到垂直和水平一致的留白。



## 2. 代码案例

代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - margin 与 padding 设置百分比</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 640px;
      height: 240px;
      border: 1px solid #f00;
    }
    .wrap p {
      float: left;
      width: 18%;
      /* 实际显示效果 */
      height: 83%;
      margin: 8.5% 1%;
      /* 我们希望显示效果 */
      width: 116px;
      height: 200px;
      margin: 20px 6px; /*
      background: #ccf;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <p>HTML5 布局之路</p>
    <p>HTML5 学堂</p>
    <p>独行冰海</p>
    <p>梦幻雪冰</p>
    <p>分享知识,共同进步</p>
  </div>
</body>
</html>
```

(注：在移动端开发时，类名为 wrap 的 div 应当设置百分比，或者直接自适应屏幕，在此为了便于理解盒模型问题，设置成了固定的 640 像素×240 像素。)





我们希望能够实现图 14.6 中的效果(或者说在书写代码的时候是这样的想法)。

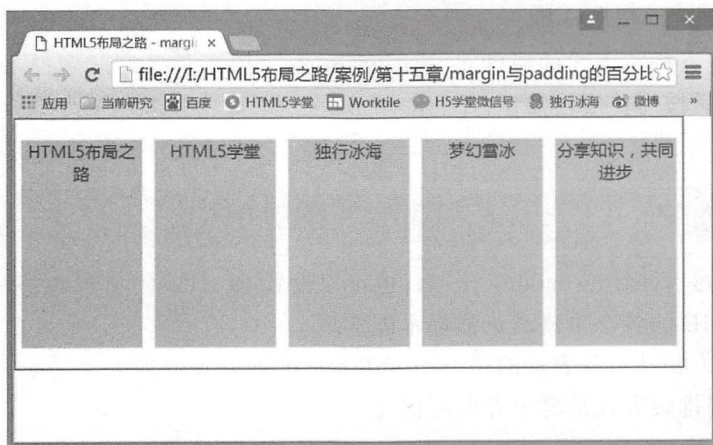


图 14.6 margin 与 padding 设置百分比时,期望的显示效果

但是当调试代码的时候,却发现展示的效果如图 14.7 所示。

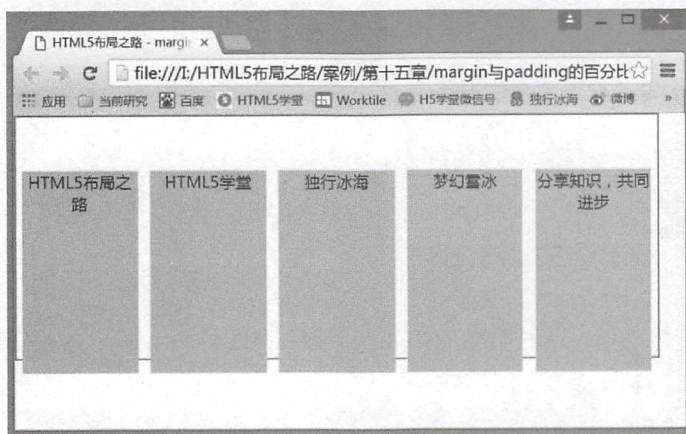


图 14.7 margin 与 padding 设置百分比时,实际的显示效果

margin 4 个方向的百分比均是基于父级的宽度(640px)进行计算,也就是说,每个 p 标签的宽度为 116px(18%)、高度为 200px(83%)、横向 margin 值左右各 6px(1%)、纵向 margin 值上下各为 54px(8.5%)。如此计算,一个 p 元素盒模型的大小为 128px×308px,明显超出了 128px×240px 的预期大小。

### 3. CSS 排版原理

CSS 的基本模型着重于“排版”的需求,因此水平和垂直方向其实并不是同等权重的,当文字是横向排列时,就采用水平方向上的排版模式,排版默认是水平宽度一定,垂直方向上可以无限延展。当文字是纵向排列时,就采用垂直方向上的排版模式,排版默认是垂直高度一定,水平方向上可以无限延展。

在使用百分比设置内外边距时,文字书写方向决定 margin/padding 按照 height 还是 width 计算。当文字是横向排列时,内外边距的百分比参照父级元素的 width 来计算(文字



纵向排列时,参照 height)。

#### 4. 额外拓展——纵向排版

控制一个元素排版方式的属性是 writing-mode。由于其兼容问题严重,在日常开发中不会使用,因此只作为扩展性知识。

代码范例:

```
writing-mode: tb-lr;           //先上下再左右(即纵向排版)
```

需要注意的是, writing-mode: tb-lr; (也可以设置 tb-rl、lr-tb 等属性值) 只有 IE8+ 浏览器兼容,其他非 IE 的各个主流浏览器都不能实现。

t 表示的是 top(上), b 表示的是 bottom(下); l 表示 left(左), r 表示 right(右)。此处的代码用于控制排版方式是哪个方向最优先。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 纵向排版</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 640px;
      height: 240px;
      border: 1px solid #f00;
    }
    .wrap p {
      float: left;
      width: 18%;
      height: 83%;
      margin: 8.5% 1%;
      background: #ccf;
      text-align: center;
      writing-mode: tb-lr;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <p>HTML5 布局之路</p>
    <p>HTML5 学堂</p>
    <p>独行冰海</p>
    <p>梦幻雪冰</p>
    <p>分享知识,共同进步</p>
  </div>
</body>
</html>
```





实现的效果图如图 14.8 所示。

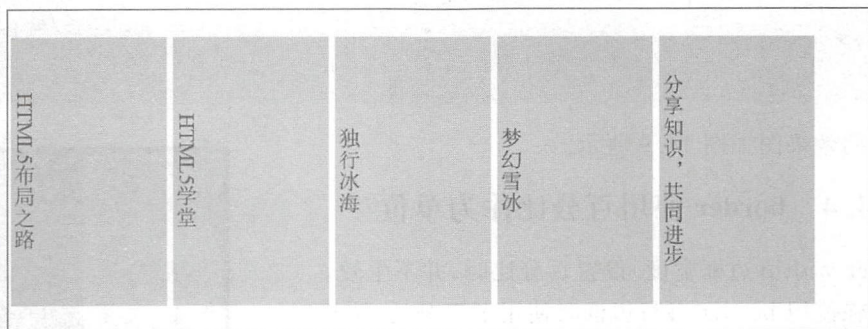


图 14.8 IE 浏览器纵向排版效果

### 14.4.3 height 使用百分比作为单位

height 设置为百分比时,是按照父级的高度进行计算的。

代码范例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - height 设置百分比</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 210px;
      height: 200px;
      border: 5px solid black;
    }
    .percent {
      float: left;
      width: 100px;
      height: 75%;
      background: #ccc;
    }
    .px {
      float: right;
      width: 100px;
      height: 150px;
      background: #39f;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "percent">高度使用百分比设置</div>
```



```

    <div class = "px">高度使用像素值设置</div>
  </div>
</body>
</html>

```

实现的效果图如图 14.9 所示。

#### 14.4.4 border 使用百分比作为单位

border-width(边框宽度)设置百分比时,并不生效。

如果在使用 border 书写边框时使用了百分比,边框整体不会被解读。

如果使用分写的方式(border-width、border-style、border-color)书写边框样式,且以百分比作为单位设置了border-width,border-width并不会被解读,但其他两行代码会生效,此时会依照浏览器默认边框宽度(3px)来显示。

代码实例:

```

<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - border 设置百分比</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box {
      width: 200px;
      height: 100px;
      border: 5px solid #39f;
    }
    .con {
      width: 50px;
      height: 50px;
      border - width: 10 % ;
      border - style: solid;
      border - color: # f00;
      /* border: 10 % solid # f00; */
    }
  </style>
</head>
<body>
  <div class = "box">
    <div class = "con"></div>
  </div>
</body>
</html>

```

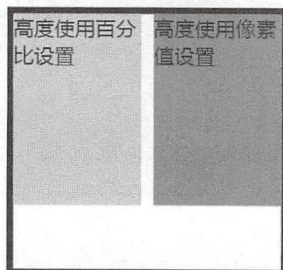


图 14.9 height 使用百分比的显示效果





border 使用百分比的显示效果如图 14.10 所示。

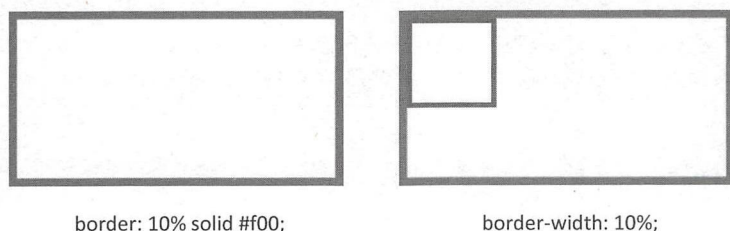


图 14.10 border 使用百分比的显示效果

### 14.4.5 line-height 使用百分比作为单位

line-height 设置百分比时,是针对 em 进行的,换句话说,行高的基数是元素本身的字体尺寸,当前元素默认字体大小是 16px,那么设置行高是 100%也就是 line-height: 16px 的含义。

代码实例:

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - line-height 设置百分比</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 420px;
      height: 80px;
      border: 5px solid black;
    }
    .percent {
      float: left;
      width: 200px;
      height: 40px;
      font-size: 20px;
      line-height: 200 % ;
      background: #ccc;
    }
    .px {
      float: right;
      width: 200px;
      height: 40px;
      font-size: 20px;
      line-height: 40px;
      background: #39f;
    }
  </style>
</head>
```



```
<body>
  <div class = "wrap">
    <div class = "percent">行高使用百分比设置</div>
    <div class = "px">行高使用像素值设置</div>
  </div>
</body>
</html>
```

显示效果如图 14.11 所示。



图 14.11 line-height 使用百分比的显示效果



## 14.5 CSS3 新增度量单位

### 14.5.1 新度量单位

- (1) ch——字符 0(零)的宽度；
- (2) rem——根元素(HTML 元素)的 font-size；
- (3) vw——viewport width, 视窗宽度, 1vw 等于视窗宽度的 1%；
- (4) vh——viewport height, 视窗高度, 1vh 等于视窗高度的 1%；
- (5) vmin——vw 和 vh 中较小的那个。

所有的这些单位,均为相对度量单位。在这些单位当中,vw、vh、vmin 是针对移动设备的,如果视窗大小发生变化,这三个值也会跟着相应地变化。rem 单位可以理解为 px+em 的综合体,比 px 灵活,比 em 更精确。

### 14.5.2 rem 与 em

em: 很不错的相对度量单位,但是计算起来实在费劲。em 是相对于当前对象内文本的字体大小进行设置。如当前对行内文本的字体尺寸未被人为设置,则相对于浏览器的默认字体尺寸。换句话说,如果当前 div 的字体大小是 12px,那么 1em 就是 12px,如果 div 字体大小是 24px,1em 就是 24px。但是每一级别都需要如此计算,实在是让人头疼。

rem,应该说让各个开发者眼前一亮,rem 虽然也是和字体相关的相对度量单位,但是,它要比 em 使用起来更方便,rem 是相对于根元素的字体大小进行设置,如果 HTML 元素中的字体大小设置为 24px,那么针对任意 HTML 内的元素设置 1rem,均表示的是 24px,大大节省了计算字体大小的时间。

rem 能够和 JS 配合,很好地解决移动端的各种像素大小问题——通过 JS 获取设备宽度,然后根据设备宽度调整 HTML 元素这个根元素的字体大小,在 HTML 元素中的所有元素,均使用 rem 相对度量单位。

rem 的支持程度: IE9 及以上所有浏览器,安卓 2.1 以上版本,iOS 4.0 以及以上版本的



Safari(换句话说,IE6~IE8 不兼容,但完全可以广泛应用在移动端)。

### 14.5.3 vw 与 vh

vw、vh 是针对移动设备的,如果视窗大小发生变化,这两个值也会跟着相应地变化。应该说,这两种单位非常好用,在项目中,采用 vw 和 vh 能够很方便地处理各个设备的布局。

vw 和 vh 当前的兼容情况不容乐观(兼容情况详见图 14.12)。不过,相信在不久的将来,随着移动端的迅速发展,vw 和 vh 会逐渐被广泛使用。

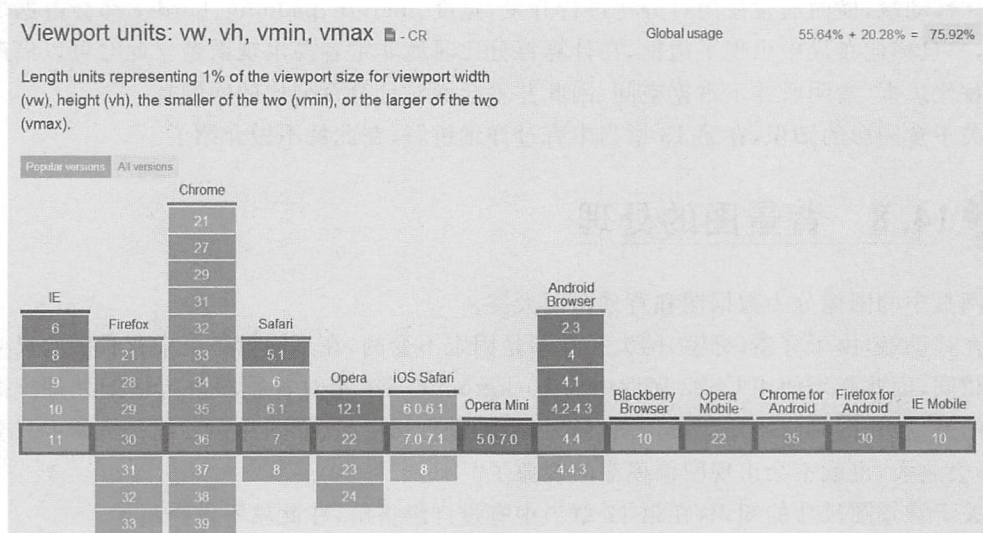


图 14.12 vw、vh 的浏览器支持程度



## 14.6 字体处理不容小觑

### 14.6.1 美工图设计的基准字体

当前为了让前端能够书写出兼容各个分辨率的代码,设计师在做移动端设计图的时候,通常会出 750px 或以上宽度的图。如果你的设计师拿着一张 320px 宽度的 PSD 文件给你,而老板要求你去制作兼容各个分辨率的移动端代码。本书建议:“让设计师返工”。

在基准字体方面也是有相应要求的。靠谱的设计师不需要你去跟他沟通,因为他们本身就懂移动端的相关设计。如果设计师对移动端的设计不了解,很建议提前与他沟通一下,以防止大像素的设计图中出现了 12px、16px 大小的字体。

那么在某一个分辨率下,最小的字体应该是多少呢?

使用 rem 进行制作的时候,通过 JS 的控制,rem 是随设备宽度变化而变化的。如果想在 320px 的设备宽度下,保证 12px 的字体大小,在宽度 1080px 的设备上,字体最小为:  $12/320 \times 1080 = 40.5$ 。也就是最小字体应当设置为 42px(字体大小应为偶数且为整数)。对于其他宽度的设备,也根据这个比例进行换算,之后向上进位即可。

### 14.6.2 移动端网络字体使用更加频繁

随着网页的发展,网页中出现了越来越多的字体种类,原有的微软雅黑以及宋体早就无法满足设计的需要,在移动端,网络字体得到了更为广泛的使用。

关于网络字体的知识,在第7章当中有过详细讲解,在此就不做介绍了。



## 14.7 盒阴影的妙用

在移动端,横向通常使用百分比进行开发,宽度、margin、padding、border 都会占据像素大小,一旦横向布局中出现了边框,在计算百分比时就非常容易出现误差。此时可以借助盒阴影替代边框(盒阴影并不占据空间,因此并不会面对计算百分比的问题)。

关于盒阴影的知识,在第13章当中有过详细讲解,在此就不做介绍了。



## 14.8 背景图的处理

网页中的图像分为数据图和背景图两大类。

在移动端,由于容器(标签)的大小并不是固定不变的,在不同设备当中背景图的尺寸也并不相同,因此必须使用 CSS3 属性中的 background-size 来对背景图进行适配处理。需要注意的是,在切图时应当按照设计图“最大尺寸”进行切图,这样图片在其他分辨率中只会缩小,不会放大,也就不会出现图像模糊的现象了。

关于背景图尺寸的知识,在第12章当中有过详细讲解,在此就不做介绍了。



## 14.9 使用 JS 配合 rem 让页面适应各个分辨率

在此以设计图的顶部为例,来从头到尾讲解一下如何实现“分辨率自适应”页面的开发。包括在开发当中需要使用到的 JS,在这里也会提及。

由于此处重点在于介绍开发流程和开发用的 JS 文件,因此案例会比较简单。对于页面类的较大实例开发,请查看第19章。

### 14.9.1 Step1 查看设计图,确定需要兼容的分辨率

设计图分析如图14.13所示。

设计图的内容区宽度为1080px。这也就意味着,在开发时,需要兼容的最大分辨率为1080px,最小的为320px。

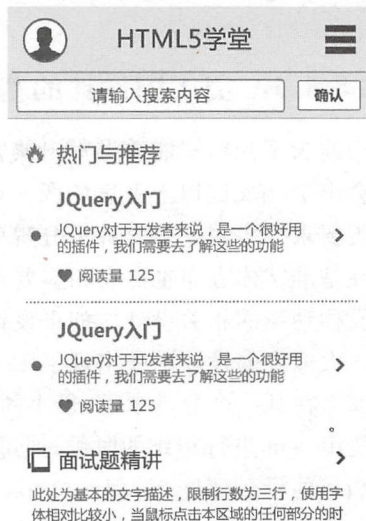


图 14.13 移动端案例要实现的设计图



## 14.9.2 Step2 调整视口

在 HTML 文件当中添加如下代码：

```
<meta name = "viewport" content = "width = device - width, user - scalable = no">
```

当前代码效果：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 移动端开发实例</title>
  <meta name = "viewport" content = "width = device - width, user - scalable = no">
  <link rel = "stylesheet" href = "../css/reset.css">
</head>
<body>
  <div class = "wrap"></div>
</body>
</html>
```

代码解析：

由于使用不同设备打开网页时，宽度均有所不同，所以不能将视口设置为固定值，应当为  $\text{width} = \text{device} - \text{width}$ ，即将视口设置为当前设备的宽度。

## 14.9.3 Step3 确定设计图的最小字体

浏览器(部分)能够显示的最小字体为 12px，或者说能够使用的最小字体为 12px。当移动端页面宽度为 320px，要保证最小字体为 12px，那么在 1080px 的设计图中，最小字体应该是多少呢？

$12 / 320 \times 1080 = 40.5$ 。由于字体必须为整数，且不能为奇数，因此在 1080px 的设计图当中，最小字体应当为 42px。

为 HTML 标签设置基准字体，font-size: 42px。

```
<style>
  html {
    font-size: 42px;
  }
</style>
```

## 14.9.4 Step4 按照设计图的像素进行开发

布局分析如图 14.14 所示。

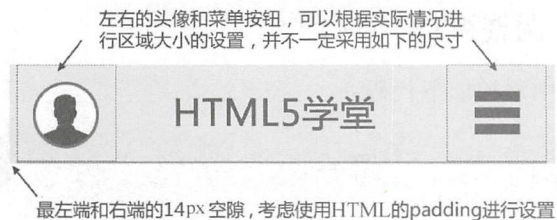


图 14.14 移动端案例顶部导航布局分析

代码范例：

```
<!doctype html >
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 移动端开发实例</title>
  <meta name = "viewport" content = "width = device - width, user - scalable = no">
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    html {
      font - size: 42px;
    }
    body {
      max - width: 1080px;
    }
    .wrap {
      padding: 0 14px;
      background: # eee;
    }
    nav {
      height: 195px;
      box - shadow: inset 0 - 5px # a6a6a6;
    }
    nav span {
      float: left;
      width: 121px;
      height: 121px;
      padding: 35px 38px;
    }
    navimg {
      display: block;
      width: 100 % ;
      height: 100 % ;
      box - shadow: 0 0 6px # 727272;
      border - radius: 50 % ;
    }
    nav h1 {
      float: left;
      width: 658px;
```



```
font-size: 72px;
text-align: center;
line-height: 195px;
}
nav strong {
float: right;
width: 197px;
height: 191px;
background: url('../images/menu.png') 0 0 no-repeat;
background-size: 100%;
}
</style>
</head>
<body>
<div class="wrap">
<nav>
<span><imgsrc="../images/HTML5 学堂-头像.jpg" alt="" title=""></span>
<h1>HTML5 学堂</h1>
<strong></strong>
</nav>
</div>
</body>
</html>
```

代码解析：

顶部导航中,左侧的头像、中间的标题、右侧的菜单按钮,采用的是浮动布局。当然也可以采用定位的布局方式。

对于头像周围的“描边”效果,标题下面的浅灰色虚线,都可以使用盒阴影。

对于背景图,要使用 background-size 进行处理。

设置最大宽度的作用,是防止页面尺寸超出设计图最大宽度。

14.9.5 Step5 使用百分比和 rem 替换 px

代码效果对比如表 14.1 所示。

表 14.1 绝对度量单位与相对度量单位实现代码案例对比

使用固定像素	使用百分比和 rem
html { font-size: 42px; } body { max-width: 1080px; } .wrap { padding: 0 14px; background: #eee; }	html { font-size: 42px; } body { max-width: 1080px; } .wrap { padding: 0 1.3%; background: #eee; }

续表

使用固定像素	使用百分比和 rem
<pre>nav {     height: 195px;     box-shadow: inset 0 - 5px # a6a6a6; } nav span {     float: left;     width: 121px;     height: 121px;     padding: 35px 38px; } navimg {     display: block;     width: 100 %;     height: 100 %;     box-shadow: 0 0 0 6px # 727272;     border-radius: 50 %; } nav h1 {     float: left;     width: 658px;     font-size: 72px;     text-align: center;     line-height: 195px; } nav strong {     float: right;     width: 197px;     height: 191px;     background: url('../images/menu.png') 0 0 no -repeat;     background-size: 100 %; }</pre>	<pre>nav {     height: 4.64rem;     box-shadow: inset 0 - 0.12rem # a6a6a6; } nav span {     float: left;     width: 11.2 %;     height: 2.88rem;     padding: 0.83rem 3.52 %; } navimg {     display: block;     width: 100 %;     height: 100 %;     box-shadow: 0 0 0 0.14rem # 727272;     border-radius: 50 %; } nav h1 {     float: left;     width: 60.93 %;     font-size: 1.71rem;     text-align: center;     line-height: 4.64rem; } nav strong {     float: right;     width: 18.24 %;     height: 4.55rem;     background: url('../images/menu.png') 0 0 no-repeat;     background-size: 100 %; }</pre>

代码解析：

水平方向的值,将具体像素调整为百分比。百分比的计算是根据父级的内容区宽度进行计算的。

例如,父级宽度为 1080px,子级元素为 197px,那么子元素转换为百分比为: 197/1080×100%=18.24%。需要注意的是百分比根据父级计算,当标签结构级别不同时,计算公式中的“分母”也有所不同,在开发时这个地方很容易出现问题,请务必注意。

垂直方向的值,将具体像素调整为 rem,与水平方向相比,垂直方向的计算就比较简单。

例如,行高为 195px,HTML 标签当前的字体大小为 42px,将行高转换为 rem 单位,即 195/42=4.64rem。

额外备注：

在计算百分比时,如果希望计算 197(子级)在 1080(父级)的百分比时,建议使用



19 700/1080 的方式,而不是计算 197/1080 之后再自行调整百分比,这样的建议,主要是由小数点会涉及精准度的问题。

Sublime 工具提供给了快速计算方法,在编辑器当中输入: 19 700/1080 之后,按下 Shift+Ctrl+Y 组合键,即可进行计算。但是需要注意的是,不要在四则运算符的左右添加空格。

### 14.9.6 Step6 使用 JS 控制基准字体

到当前步骤,我们完成了 1080px 下的设计图(导航部分),使用百分比和 rem 作为单位,让网页能够随着设备宽度和基准字体大小的变化而发生变化。

设备宽度会随着加载设备的不同而有所不同,但是基准字体大小如何变化呢? 此处就需要借助 JS 来动态地改变基准字体大小。

JS 文件(JS 文件名为 changeFont.js):

```
$ (function(){
    $ (window).resize(infinite);
    function infinite() {
        varhtmlWidth = $ ('html').width();
        if (htmlWidth>= 1080) {
            $ ("html").css({
                "font-size" : "42px"
            });
        } else {
            $ ("html").css({
                "font-size" : 42 / 1080 * htmlWidth + "px"
            });
        }
    }
    }infinite());
});
```

**代码解析:**

这段代码是 JavaScript(JS)代码,基本功能是:

检测当前网页的宽度,如果当前页面宽度大于 1080px,则为 HTML 标签设置 42 号字体(网页的基准字体就是 42px 了),如果当前页面宽度小于 1080px,则通过等比例换算,得出当前像素值状态下的基准字体,计算公式为:

当前分辨率基准字体=最大分辨率下的基准字体(42px)/最大分辨率(1080px)  
×当前页面分辨率(通过 JS 获得页面宽度)

**额外注意:**

该段代码基于 zepto.js 框架,因此如果需要使用,请先下载并引入 zepto.js。

zepto.js 下载地址:

<https://github.com/madrobby/zepto> 或者 <http://zeptojs.com/>

也可以在本书提供的电子资料当中下载(如果希望获取最新版本,还请在官方地址当中下载)。

## JS 文件引入的代码实例：

```
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 移动端开发实例</title>
  <meta name = "viewport" content = "width = device - width, user - scalable = no">
  <link rel = "stylesheet" href = "../css/reset.css">
  <script src = "../js/zepto.js"></script>
  <script src = "../js/changeFont.js"></script>
</head>
```

注意：由于网页是自上而下加载的，zepto.js 的代码要书写在前面。

## 完整的代码实例：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 移动端开发实例</title>
  <meta name = "viewport" content = "width = device - width, user - scalable = no">
  <link rel = "stylesheet" href = "../css/reset.css">
  <script src = "../js/zepto.js"></script>
  <script src = "../js/changeFont.js"></script>
  <style>
    html {
      font-size: 42px;
    }
    body {
      max-width: 1080px;
    }
    .wrap {
      padding: 0 1.3%;
      background: #eee;
    }
    nav {
      height: 4.64rem;
      box-shadow: inset 0 -0.12rem #a6a6a6;
    }
    nav span {
      float: left;
      width: 11.2%;
      height: 2.88rem;
      padding: 0.83rem 3.52%;
    }
    navimg {
      display: block;
      width: 100%;
      height: 100%;
      box-shadow: 0 0 0 0.14rem #727272;
```



```
border-radius: 50%;
}
nav h1 {
  float: left;
  width: 60.93%;
  font-size: 1.71rem;
  text-align: center;
  line-height: 4.64rem;
}
nav strong {
  float: right;
  width: 18.24%;
  height: 4.55rem;
  background: url('../images/menu.png') 0 0 no-repeat;
  background-size: 100%;
}
</style>
</head>
<body>
  <div class="wrap">
    <nav>
      <span><imgsrc="../images/HTML5 学堂-头像.jpg" alt="" title=""></span>
      <h1>HTML5 学堂</h1>
      <strong></strong>
    </nav>
  </div>
</body>
</html>
```

# 第15章

## 转战移动端(下)

### ——响应式&移动端的探索



#### 15.1 响应式布局

##### 15.1.1 理解响应式布局

###### 1. 什么是响应式布局

响应式布局,称为 Responsive Web Design。它是将已有的开发技巧(弹性网格布局、弹性图片、媒体和媒体查询)整合起来,针对任意设备对网页内容进行“完美”布局的一种显示机制。

简言之,是一个网站能够兼容多个终端(手机、Pad、计算机)的布局方法,而不需要为每个终端书写一套特定版本的代码。

###### 2. 响应式布局的主要发展历程

在最初互联网向移动互联网发展时,如果在移动端采用传统的网页展示,会造成很差的用户体验,此时通常会为移动端单独开发一套网站。随着设备分辨率种类的增加,如 320 宽度、640 宽度、1024 宽度、1366 宽度等。此时如果针对每一个设备大小进行一套网站的开发,工作量巨大,相对比较消耗成本。2010 年 5 月,Ethan Marcotte 提出响应式的概念;

2012 年,响应式布局开始在行业内逐渐流行,关于响应式布局的框架也开始频繁出现;

2013 年,被称为响应式 Web 设计年。TNW 在 2013 年年初发布的“2013 年 10 大网页设计趋势”当中,响应式布局“首当其冲”。

##### 15.1.2 响应式布局的优劣势

###### 1. 响应式布局的优势

多终端视觉和操作体验风格统一;

兼容当前及未来新设备;

响应式 Web 设计中的大部分技术可以用在 WebApp 开发中;

节约了开发成本,维护成本也降低很多。



## 2. 响应式布局的劣势

IE8-等低版本浏览器存在兼容问题;

移动带宽流量有一定的问题,由于响应式布局的网站代码中包含各个终端的代码,因此,与手机定制网站相比较,流量较大;

代码累赘,会出现隐藏无用的元素,加载时间加长;

兼容各种设备工作量大。

### 15.1.3 响应式布局的核心技术

#### 1. 最核心的三点技术

通过 meta 调整视口;

通过媒体查询,为不同设备加载相应样式;

相对单位替换绝对单位。

#### 2. 相对单位替换绝对单位是什么意思

宽度和高度不以固定单位 px 存在,而是采用百分比、rem 进行设置;字体大小使用相对度量单位 em 或 rem 代替绝对单位 px。

#### 3. 样式的书写顺序

由于响应式布局的代码当中,包括各个终端的代码。因此,建议先书写基准样式(各个视图宽度的共同样式)、再书写移动网站样式、Pad 版样式,最后书写桌面网站样式。



## 15.2 媒体查询

### 15.2.1 什么是媒体查询

媒体查询(CSS3 Media Query),又叫媒介查询,主要作用是根据网页的具体条件,告知浏览器如何为指定的视图宽度渲染页面。

媒体查询是一些逻辑表达式,用于计算用户浏览器中媒体特性的当前值,如果媒体查询表达式计算的结果为 true,则应用其所包含的 CSS 规则。

### 15.2.2 媒体查询书写方法

#### 1. 有条件的应用样式

代码实例:

```
<style>
  @media all and (min-width:500px) { ... }
  @media (orientation: portrait) { ... }
</style>
```

代码解析:

第一行媒体查询代码指的是:为宽度大于等于 500px 的设备设置样式。

第二行媒体查询代码指的是：为纵屏状态（可见区大于或等于宽度）下的移动端设备设置样式。

## 2. 有条件的加载样式表

代码实例：

```
<head>
  <link rel = "stylesheet" href = "wide.css" media = "screen and (min-width:1024px)" />
  <link rel = "stylesheet" href = "mobile.css" media = "screen and (max-width:320px)" />
</head>
```

代码解析：

第一行媒体查询代码指的是：为宽度大于等于 1024px 的设备，加载 wide.css 文件。

第二行媒体查询代码指的是：为宽度小于等于 320px 的设备，加载 mobile.css 文件。

### 15.2.3 常见媒体类型

braille	触觉反馈设备
embossed	盲文印刷设备
handheld	小型或手持设备
print	打印机
projection	投影图像，如幻灯
screen	计算机显示器
speech	语音合成器
tty	打字机
tv	电视类

### 15.2.4 关于媒体的特性

每个人都有自己的属性，例如，年龄、性别、身高、体重等。作为媒体（计算机、手机、电视、电话），也有它们的特性。CSS3 针对所有的媒体均定义了 width、color 等媒体特性。作为媒体特性本身，其作用并没有什么，但是对于开发者来说，可以通过检测媒体的特性确定不同的媒体，从而执行相应处理。

常见媒体特性如下。

width	布局视口宽度
height	布局视口高度
device-width	设备屏幕宽度
device-height	设备屏幕高度
orientation	设备方向（portrait 表示纵屏，landscape 表示横屏）
aspect-ratio	视口宽高比
device-aspect-ratio	屏幕宽高比
resolution	设备像素密度





## 15.3 让移动端开发变得更好——关于高清屏幕

### 15.3.1 高清分辨率

在 iPhone4 开始,手机屏幕就进入了高清时代。  
何为高清屏幕呢? 比较 iPhone3g 和 iPhone4,如表 15.1 所示,就不难理解。

表 15.1 iPhone3g 和 iPhone4 设备属性对比

设备类型	iPhone3g	iPhone4
设备物理宽度	320×480	320×480
设备分辨率	320×480	640×960

能够看出,虽然两种设备的物理宽度都是一样的,但是实际分辨率却差了 4 倍(横向两倍、纵向两倍)。对于 iPhone3g 设备来说,一个物理像素对应一个分辨率像素点;而 iPhone4 设备,一个物理像素点对应 4 个分辨率像素点( $640\times960/320\times480$ ),此时,每个物理像素点的图像就会变得更加精确,而且放大之后(两倍以内)图像依旧清晰。

### 15.3.2 此前移动端开发存在的一些不足

第 14 章中讲解了不同分辨率适配的方法:将视口设置为屏幕物理宽度,之后进行基准像素的调整,调整公式如下:当前分辨率基准字体=最大分辨率下的基准字体(42px)/最大分辨率(1080px)×当前页面分辨率(通过 JS 获得页面宽度)。

这种方法当中,物理宽度 320px 的设备,网页宽度即为 320px;物理宽度 480px 的设备,网页宽度即为 480px。如果希望页面能够以高清来显示(即在物理宽度 320px 的设备当中,网页宽度为 640px,物理宽度 480px 的设备当中,网页宽度为 960px),就出现了问题。

### 15.3.3 按照高清分辨率解读的“想法”

#### 1. flexible.js

淘宝团队基于高清分辨率的需求,开发了一款 flexible.js 框架。  
对比 flexible.js 与此前讲过的实现方法,其实变化的并不是很多,如表 15.2 所示。

表 15.2 实现移动端网页的基本方法与 flexible.js 方法的对比

此前的书写方法	flexible.js
视口大小为屏幕物理宽度	视口大小为屏幕物理宽度
根据设计图、实际屏幕宽度,调整基准字体	根据设计图、实际屏幕宽度,调整基准字体
缩放比例为 1	根据不同设备的设备像素比例,调整不同的缩放比例

比较两者,能够发现 flexible 只是在缩放比例的位置进行了控制。而且,缩放比例的设置并不需要开发工程师来进行操作,flexible.js 会代替开发工程师完成这个工作。

#### 2. 关于设备像素比例

flexible.js 会获取到“设备的物理宽度”和“设备像素比例”,之后根据这两者计算出设

备的“可用布局宽度”。例如,iPhone4S 当中,设备物理宽度为 320px,设备像素比例为 2,那么可用布局宽度为 640px。初始缩放比例就应当设置为 0.5 (initial-scale=0.5),使 iPhone4S 按照 640px 渲染。

简单地说,不同设备有不同的比例,这个是由设备生产商决定的,设备像素比例 $\times$ 初始缩放比例=1。根据此公式即可计算出初始缩放比例。

### 15.3.4 flexible.js 的用法

#### 1. 下载地址

官方地址: <https://github.com/amfe/lib-flexible>。

也可以在本书提供的电子资料当中下载(如果希望获取最新版本,还请在官方地址当中下载)。

#### 2. 使用步骤

flexible.js 的使用步骤与 14.9 节介绍的使用步骤基本一致(只是在最后一步有所不同)。

- (1) 查看设计图,确定需要兼容的分辨率;
- (2) 调整视口(虽然 flexible.js 会自动添加视口的 meta 信息,但是还是建议先进行书写,不然会引发部分安卓机器的问题,此处的问题请详见下 15.3.5 节);
- (3) 确定设计图的最小字体;
- (4) 按照设计图的像素进行开发;
- (5) 使用百分比和 rem 替换 px;
- (6) 使用 flexible.js,调整视口和基准字体。

假设计图为 960px,且确定基准字体大小为 36px,此时在 flexible.js 中修改如下代码(该代码只是 flexible 的一部分)。

```
function refreshRem(){
    var width = docEl.getBoundingClientRect().width;
    if (width / dpr > 960) {
        width = 960 * dpr;
    }
    var rem = width / (960 / 36);
    docEl.style.fontSize = rem + 'px';
    flexible.rem = win.rem = rem;
}
```

如果希望了解 flexible.js 的工作原理,可以查看如下的原理解析。

flexible.js 首先获取设备的宽度,存储于 width 变量当中,通过获取设备像素比例计算设备缩放比例,并存储于 dpr。(第 2 行代码,dpr 的设置 flexible.js 的其他地方。)

针对最大宽度进行控制,当设备宽度除以设备像素比例大于设计图的最大宽度时,就让宽度等于设备缩放比 $\times$ 设计图最大宽度(第 3 行~第 5 行代码);否则就使用设备宽度。

基准字体大小的设置,根据“当前设备的布局宽度”、“PSD 原图宽度”、“PSD 图字体基准值”,计算出“当前设备页面基准字体大小”(第 6 行,rem 的设置),之后将其设置给



HTML 元素,从而控制整个页面中 rem 控制的数值。(最后两行)

### 15.3.5 flexible 框架使用的注意事项

虽然 flexible 框架会自动添加视口的 meta,但如果页面中最初没有设置关于视口的 meta 信息,在安卓的部分手机下,边框(border)、阴影的解析会出现变粗的问题。因此,建议在开发时先手动添加关于视口的 meta 信息。



## 15.4 让移动端开发变得更快——固定像素的实现方法

虽然横向百分比、纵向 rem 的代码实现方法,很好地解决了移动端页面开发的需求。但是这种实现方法,需要在原有绝对单位的基础之上再进一步操作。作为从最初就使用绝对单位(px)书写前端页面的开发者,有谁不希望能够直接使用 px 单位实现移动端页面呢?

网易开发团队,研发了一款新框架 MetaHandler.js,就是通过固定像素(绝对单位)实现移动端的开发。

### 15.4.1 MetaHandler.js

#### 1. 下载地址

官方地址:

<https://github.com/unbug/generator-webappstarter/blob/master/app/templates/app/src/util/MetaHandler.js>

也可以在本书提供的电子资料当中下载(如果希望获取最新版本,还请在官方地址当中下载)。

#### 2. 基本原理

该框架主要利用了视口中的缩放值,在视口设置当中,可以通过调整缩放值,控制网页的整体缩放。

例如,使用固定像素 960px 书写的页面,之后检测到加载网页的设备宽度为 640px,此时将视口缩放为原来的 66.67%(640/960)。

#### 3. 具体使用步骤

(1) 按照 PSD 图的实际宽度(即兼容的最大分辨率),使用像素单位进行页面的基本书写(和写 PC 端页面一样)。

(2) 在页面顶部添加 meta,设置视口信息,将 width 设置为 PSD 图的实际宽度(即制作页面时的实际宽度),但是请不要设置缩放值。如:

```
<meta content = "target - densitydpi = device - dpi,width = 1080" name = "viewport">
```

(3) 修改 MetaHandler.js 文件的最后配置代码。在执行 fixViewportWidth 函数时,传入相应的像素值作为参数即可,上一步的案例当中应当传入 1080,即:

```
opt.fixViewportWidth(1080);
```

#### 4. 网页案例

鉴于很多人没有接触过 JavaScript, 因此, 在此给出 HTML 文件的结构实例。

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - metahandler.js 案例</title>
  <meta content = "target - densitydpi = device - dpi,width = 1080" name = "viewport">
  <link rel = "stylesheet" href = "../css/reset.css">
</head>
<body>
  <div>网页的具体内容部分</div>
  <!-- 引入 MetaHandler -->
  <script type = "text/javascript" src = "../js/MetaHandler.js"></script>
</body>
</html>
```

修改 metahandler.js 中的最后一句代码:

```
opt.fixViewportWidth(1080);
```

### 15.4.2 框架当前还存在的问题

#### 1. 谷歌浏览器特殊的留白情况

在使用谷歌浏览器进行浏览时, 如果在纵屏转向横屏状态之后刷新, 页面会再缩小, 成为居中显示状态, 之后再从横屏切换到竖屏时, 左右也会留白(此时再刷新能重新恢复正常)。

如果在从纵屏转向横屏时没有刷新, 并不会触发这个问题。

横屏后再刷新的效果如图 15.1 所示。



图 15.1 横屏后再刷新的效果

从横屏恢复到纵屏的状态如图 15.2 所示。

由于很少有人会在旋转到横屏之后再刷新页面, 加之目前移动端使用谷歌浏览器的人





图 15.2 从横屏恢复到纵屏的状态

较少(国内),所以个人感觉这个 bug 是可以忽略的。另外,对于横屏转回到纵屏的时候,可以进行设备方向检测,然后再触发一次页面刷新,就不会出现纵屏上的问题了。

## 2. 文字大小解读有误

在魅族 4 当中的内置浏览器进行测试时,当手指滑过部分文字的时候,文字的大小会出问题。经过排查之后发现,在网页中的 a 标签会出现这个问题。当光标移动到 a 标签上的时候(即便不单击),标签的字体和行高会失效。具体原因以及解决办法尚不明确。



## 15.5 移动端兼容

### 15.5.1 CSS3 媒体查询兼容问题

IE8-浏览器并不支持 CSS3 Media Query(媒体查询)。可以使用 `css3-media-queries.js` 或者 `respond.js` 来为 IE 添加 Media Query 支持。

(1) `css3-media-queries.js` 下载地址:

<https://github.com/livingston/css3-mediaqueries-js>

(2) `respond.js` 下载地址

官方地址: <https://github.com/scottjehl/Respond>

也可以在本书提供的电子资料当中下载(如果希望获取最新版本,还请在官方地址当中下载)。

(3) 采用 css3-media-queries.js 的代码实例(respond.js 同理):

```
<!-- [if lt IE 9]>
    <script src = "http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>
<![endif]-->
```

代码解析:

版本低于 9 的 IE 浏览器(不含 IE9),加载 css3-mediaquies.js 文件,进行兼容的处理。

## 15.5.2 HTML 与 CSS 的基本兼容问题

### 1. 取消电话号码识别

具体问题:在 iPhone 上页面中的数字识别为电话号码。

开发工程师书写的初始结构代码:

```
<li>
    <h2>HTML5 布局之路</h2>
    <p>18100010001 </p>
</li>
```

在 iPhone 中代码会自动处理成如下状态:

```
<li>
    <h2>HTML5 布局之路</h2>
    <p>
        <a href = "tel:18100010001" title = "">18100010001 </a>
    </p>
</li>
```

代码变化:在 p 标签内部增加了一个 a 标签用于包含电话号码。

解决方法:在网页文件的文件头部设置如下代码即可。

```
<meta content = "telephone = no" name = "format - detection">
```

### 2. 取消电子邮箱识别

具体问题:在安卓平台手机中,页面中的邮箱信息会被识别为邮箱地址,成为可单击的链接。显示状态和上面刚讲到的 iPhone 中电话号码一样。

解决方法:在文件头部加入如下代码即可。

```
<meta content = "email = no" name = "format - detection">
```

### 3. rem 水平方向上的兼容问题

具体问题:移动端布局时,横向使用 rem(相对度量单位)时,会在部分手机浏览器当中出现问题。横向布局使用 rem 时的正常页面效果如图 15.3 所示。



华为 mate7 内置浏览器上页面效果如图 15.4 所示。



图 15.3 横向布局使用 rem 时的正常页面效果



图 15.4 华为 mate7 内置浏览器上页面效果

备注：目前只是测出华为有问题，其他测试手机没有出现问题。

解决方法：水平方向用百分比来实现。

### 15.5.3 默认样式处理

#### 1. -webkit-tap-highlight-color

具体应用：当单击一个链接或者通过 JavaScript 定义的可单击元素的时候，它就会出现一个半透明的灰色背景，开发中有时会要求去掉这个不太好看的高亮效果。

解决办法：重设这个表现，可以通过 -webkit-tap-highlight-color 为其设置任何颜色。如果想要禁用这个高亮效果，设置颜色的 alpha 值为 0 即可。

代码实例：

```
a {
    -webkit-tap-highlight-color: rgba(255,0,0,0.5);
}
```

代码解析：

设置 a 标签在单击时，存在 50%透明度的红色高亮。

备注：只出现在 iOS 系统当中(iPhone 和 iPad)。

#### 2. 浏览器文本框默认高光样式

具体应用：在浏览器当中的文本框等表单元素被“聚焦”时，会在元素周围出现高光样式，有时需要去掉这个样式。

解决办法：为 input(表单元素)的 focus 状态设置样式，将高光样式设置为“全透明”。

**代码实例：**

```
input:focus{
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    -webkit-user-modify: read-write-plaintext-only;
}
```

**代码解析：**

input:focus 表示的是“处于聚焦状态的 input 元素”。

第一句代码是 iOS、安卓 4.0 版本以下的设备去除高光的方式，第二句代码是安卓 4.0 以上版本去除高光的方式。

**3. iPhone、iPad 的按钮存在默认样式**

具体应用：最初移动端当中，出于视觉效果考虑，iPhone、iPad 的按钮均设置了默认样式。移动端飞速发展之后，很多设备当中按钮的样式都需要自行定义，此时，就需要去除默认样式。

解决办法：为三种按钮以及多行文本框设置 -webkit-appearance 属性。

**代码实例：**

```
input[type="button"], input[type="submit"], input[type="reset"] {
    -webkit-appearance: none;
}
textarea { -webkit-appearance: none;}
```

**代码解析：**

为三种按钮以及多行文本框去除默认样式。

**4. 文本框的叉号**

具体应用：在 IE10 浏览器当中，当文本框中输入内容后，在文本框的右侧会出现一个叉号(×)，出于美观考虑，很多网站都希望能够去除这个叉号。

解决办法：为 input 中的 clear 伪元素设置 display: none。

**代码实例：**

```
input::-ms-clear {
    display: none;
}
```

**代码解析：**

在文本框中的叉号(×)是以伪元素的方式显示的，因此需要针对 input 的伪元素进行设置。

::clear 是一个伪元素，-ms 是浏览器内核前缀。由于这个问题出现在 IE10 浏览器，因此需要添加 -ms 的前缀，就成了 ::-ms-clear。



## 第16章

# CSS3变形与动画



### 16.1 CSS3 二维变形

#### 16.1.1 二维变形基本语法

属性功能：

让元素发生某种变形。IE9+以及各个主流浏览器均支持。

基本语法：

```
transform: translate(100px) scale(1.2) rotate(30deg);
```

代码解析：

元素向右平移 100px,之后放大为原来的 120%,并顺时针旋转 30°。

属性值如表 16.1 所示。

表 16.1 CSS3 二维变形的属性值

值	描 述
rotate(val)	元素发生旋转
translate(val)	元素进行移动
scale(val)	放大或缩小显示元素
skew(val)	设置元素扭曲/斜切

transform 属性中分为多个变形方法(扭曲/倾斜、旋转、移动、缩放),每种变形方法有各自的属性值。因此 transform 的书写方法就不能使用“属性名称:属性值”的方法。而采用“transform:二级属性名称(属性值)”的方式,多个二级属性之间使用空格分隔,一个()中的属性值之间用逗号分隔。

注意:进行变形处理后的元素,并不会脱离文档流。

#### 16.1.2 具体变形方式语法详析

##### 1. rotate

基本语法：

```
transform: rotate(30deg);
```

**代码解析：**

该段代码表示让元素以顺时针方向旋转  $30^\circ$ 。对于元素来说,deg 表示的是旋转单位,是 degree(角度值)的缩写,默认的旋转中心为元素的中心位置,当 rotate 值为正值时为顺时针方向旋转,而负值时为逆时针方向旋转。

**1) 如果旋转角度设置大于  $360^\circ$** 

旋转角度可以大于  $360^\circ$  或小于  $-360^\circ$ ,此时,使用该角度值对 360 取余,余数作为度数的显示效果即为该角度的显示效果。

例如, $1080^\circ$  显示效果与  $0^\circ$  的显示效果相同; $-900^\circ$  显示效果与  $-180^\circ$  效果相同。 $(1080\%360=0, -900\%360=-180)$ 。取余就是数学当中获取运算之后的余数。)

**注意：**一旦涉及动画效果,则会有所不同。如: $1090^\circ$  与  $10^\circ$  从最终表现上来说都是旋转了  $10^\circ$ ,但是 1090 相当于是旋转三周之后再旋转  $10^\circ$ 。对于这方面的具体案例,在本章过渡和动画当中会有所提及。

**2) 如何改变旋转中心**

可以通过 transform-origin 命令修改元素的旋转参考中心点。

**基本语法：**

```
transform-origin: left top;
transform-origin: center center;
transform-origin: right bottom;
transform-origin: 20px 20px;
transform-origin: 50% 20px;
```

**代码解析：**

transform-origin 属性值可以使用 left、center、right; top、center、bottom 等单词,也可以使用 20% 等数值(XXX px 也可以)。第一个值表示水平方向,第二个值表示垂直方向。默认状态下,旋转中心 == center center。

不同旋转中心时,元素旋转  $15^\circ$  的显示效果如图 16.1 所示。



图 16.1 不同旋转中心点时,元素旋转  $15^\circ$  的显示效果



## 2. translate

### 基本语法：

```
translateX(100px);  
translateY(-50px);  
translate(50px);  
translate(100px, -50px)
```

### 代码解析：

translate 单位为 px, 用于设置水平方向和垂直方向的平移值, 拥有两个参数, 如果只书写一个参数(属性值), 则表示水平方向上的平移值, 而垂直方向则为 0; 如果书写两个参数(属性值), 则第一个值表示水平方向, 第二个值表示垂直方向。也可以单独针对 X 轴或 Y 轴进行平移。

属性值可以使用负值。水平方向, 默认向右为正方向, 垂直方向, 默认向下为正方向。不同属性值时, 元素平移变形的效果如图 16.2 所示。

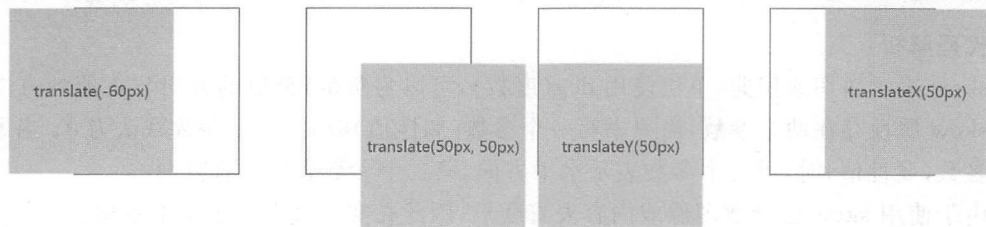


图 16.2 设置不同平移属性值时, 元素平移变形的效果

## 3. scale

### 基本语法：

```
scaleX(0.9);  
scaleY(2);  
scale(0.5);  
scale(0.5, 2)
```

### 代码解析：

scale 表示缩放, 不需要单位, 可以为小数。1 表示的是百分百的原始大小显示效果, 小于 1 的数字表示缩小, 大于 1 的数字表示放大。拥有两个参数, 如果只设置一个参数(属性值), 表示 X 轴和 Y 轴均应用该属性; 如设置两个参数(属性值), 则第一个参数表示水平方向, 第二个参数表示垂直方向。

当 scale 设置负值时, 左右、上下颠倒。

开发时需要注意, 不要设置过于大或过于小的缩放值, 放大倍数增大之后, 图像有损; 另外, 也需要注意横纵比例。

不同属性值时, 元素缩放变形的效果如图 16.3 所示。

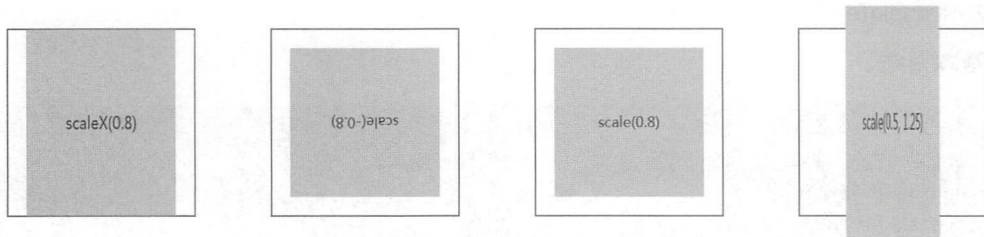


图 16.3 设置不同缩放属性值时,元素缩放变形的效果

#### 4. skew

基本语法:

```
skewX(3deg);
skewY(15deg);
skew(10deg);
skew(10deg, 10deg);
```

代码解析:

skew 表示斜切或扭曲,单位使用 deg(度数),可以为负值,负值的方向与正值的方向相反。skew 属性存在两个参数,当只书写一个参数(属性值)时,第二个参数默认为 0;当书写两个参数(属性值)时,第一个参数表示水平方向,第二个参数表示垂直方向。

由于使用 skew 会导致图像或内容失真变形,因此在实际开发当中基本不用。

不同属性值时,元素扭曲变形的效果如图 16.4 所示。

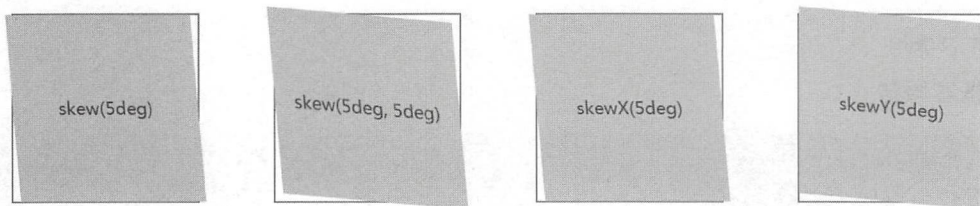


图 16.4 设置不同扭曲属性值时,元素扭曲变形的效果

### 16.1.3 变形顺序对最终结果是否会造成影响

当一个变形当中,并不仅仅存在一种变形时(如既需要旋转,又需要平移),不同变形的书写顺序有可能会对最终的显示效果造成影响。

显示效果如图 16.5 所示。

代码解析:

图 16.6 中,上面的 div 是先进进行平移,再进行旋转;下面的 div 是先进进行旋转再进行平移。这两个 div 的位置并不相同,也就意味着 rotate 和 translate 的顺序调整之后,展示出的效果不同。

变形顺序会影响最终显示效果的原理图如图 16.6 所示。



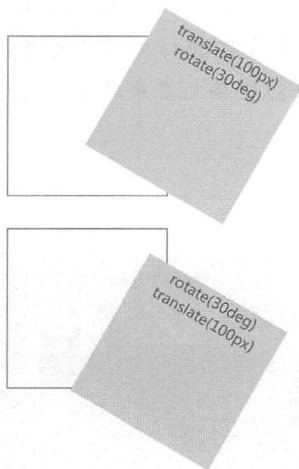


图 16.5 旋转平移——不同变形顺序的显示效果

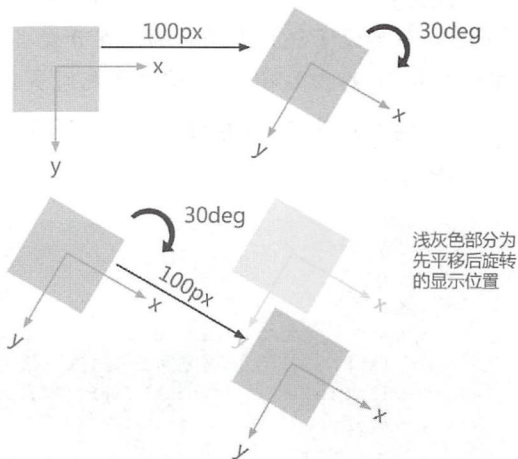


图 16.6 变形顺序影响最终显示效果原理图

原理解析：

先旋转,会引起平面坐标系随之变化,x轴跟随元素顺时针旋转了 $30^{\circ}$ ,之后按照新的x轴方向进行平移。

在日常开发当中,请调整好变形的顺序,防止因变形书写顺序而导致的问题。



## 16.2 CSS3 三维变形

### 16.2.1 如何触发三维变形

CSS3 除了为开发者提供了二维变形之外,还将动画从二维平面推动到了三维立体状态,能够实现真正的三维特效。

三维变形和二维变形一样,均使用的是 transform 属性。那么如何让浏览器知晓该段代码应当以三维变形的方式进行解读呢?

一种方式是通过语法告知浏览器“请采用三维方式进行变形处理”,另一种方式是直接使用 CSS3 三维变形的语法。

#### 1. 触发方法 1: 告知浏览器变形方式

基本语法:

```
-webkit-transform-style: preserve-3d;
```

相关说明:

IE 不支持三维变形,在移动端,绝大多数的浏览器均为 WebKit 内核,因此,在此句代码之前需要书写-webkit 的内核前缀。

该段代码需要添加给三维变形元素的父级,此时浏览器会按照三维变形的方式渲染父级以及父级内部的元素。

注意：不要为 body 元素设置 `-webkit-transform-style: preserve-3d`, 否则会对 position: fixed 定位的元素造成布局影响。在开发当中, 如果当前元素属于 body 的子级元素, 又希望应用三维变形, 则在 body 和当前元素之间多嵌套一层结构, 并为这层元素应用三维变形即可。

## 2. 触发方法 2: 直接使用 CSS3 三维变形语法

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 通过三维变形语法触发三维变形</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .box1 {
      width: 150px;
      height: 150px;
      border: 2px solid red;
    }

    .box1 div {
      height: 150px;
      background: rgba(127, 127, 127, 0.4);
      -webkit-transform: translate3d(30px, 60px, 20px) rotateX(30deg);
      transform: translate3d(30px, 60px, 20px) rotateX(30deg);
    }
  </style>
</head>
<body>
  <div class = "box1">
    <div></div>
  </div>
</body>
</html>
```

显示效果如图 16.7 所示。

备注：关于三维变形的具体属性在 16.2.3 节中会详细讲解。

### 16.2.2 Z 轴的位置

二维平面当中, 存在一个平面坐标系, 起点在元素的中心, X 轴水平向右, Y 轴向下。当变形从二维平面转到三维立体时, 标识一个位置就需要三个值, 也就是构成了立体坐标系(含 X、Y、Z)。Z 轴垂直于平面(X 与 Y 构成的平面)向上。默认状态下可以简单地理解为: 垂直于屏幕, 指向用户。但是需要注意的是, Z 轴和 X、Y 轴类似, 都会随着物体平面的变化而变化。

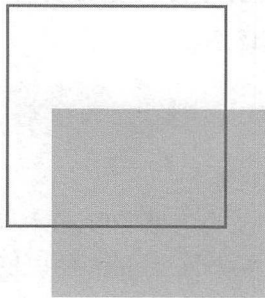


图 16.7 通过直接使用三维变形命令, 触发三维变形效果



### 16.2.3 三维变形的变形属性

#### 1. rotate

rotate 书写方式如表 16.2 所示。

表 16.2 CSS3 三维变形 rotate 书写方法

二维变形书写方式	三维变形书写方式
rotate();	rotateX(); rotateY(); rotateZ(); rotate3d(x, y, z, angle);

语法解析：

在二维变形当中，是围绕着元素的中心点进行旋转，因此只有一个参数值。

在三维变形当中，出现了三条轴(x、y、z)，旋转时可以围绕这三条轴的任意轴旋转。

rotateX 表示围绕 x 轴进行旋转、rotateY 表示围绕 y 轴进行旋转、rotateZ 表示围绕 z 轴进行旋转。

rotate3d 是三种轴旋转的缩写方式，其中，x、y、z 均为一个数字，最后一个 angle 为一个角度，如：rotate3d(1, 2, 1, 30deg)；表示的是围绕 x 轴旋转 30°，围绕 y 轴旋转 60°，围绕 z 轴旋转 30°。

#### 2. translate

translate 书写方式如表 16.3 所示。

表 16.3 CSS3 三维变形 translate 书写方法

二维变形书写方式	三维变形书写方式
translate(); translateX(); translateY();	translateX(); translateY(); translateZ(); translate3d(x, y, z);

语法解析：

在三维变形当中，可以沿着三条轴(x、y、z)的方向进行平移。translateZ 表示沿着 z 轴进行平移。

z 轴方向，垂直平面向上为正方向。

translate3d 是多方向平移的缩写方式，其中，x、y、z 均为平移的距离。

#### 3. scale

scale 书写方式如表 16.4 所示。

语法解析：

在三维变形当中，可以针对三个方向(x、y、z)进行缩放。

scaleZ 表示沿着 z 轴进行缩放。在 z 轴上的使用很少，了解即可。

scale3d 是多方向缩放的缩写方式,其中,x、y、z 均为缩放的比例数值。

表 16.4 CSS3 三维变形 scale 书写方法

二维变形书写方式	三维变形书写方式
scale();	scaleX();
scaleX();	scaleY();
scaleY();	scaleZ();
	scale3d(x, y, z);

#### 4. skew

skew 书写方式如表 16.5 所示。

表 16.5 CSS3 三维变形 skew 书写方法

二维变形书写方式	三维变形书写方式
skew();	skew();
skewX();	skewX();
skewY();	skewY();

skew 在二维、三维方面并没有变化。在实际开发当中,由于会影响内容的样式,使用相对较少。

### 16.2.4 视角

基本语法:

```
- webkit - perspective: 数字;
```

代码解析:

在默认情况下表示无限远。如果使用数字 0,表示无限远。当设置为非零的数字时,表示的是视角距离物体的像素值,当视角越接近物体,也就越容易变形。

### 16.2.5 旋转带来的问题

当一个元素能够围绕 x 或 y 轴进行旋转时,就有可能出现“显示元素背面”的状态。此时,元素的内容会透过背景显示出来(也称透视),对于一些效果,不希望“展示背面”时,内容显示出来,所以可以设置“背面不可视”的代码。

基本语法:

```
- webkit - backface - visibility: hidden;
```

### 16.2.6 关于三维变形的应用

对于三维变形,通常会与 hover 效果、CSS3 过渡(transition)、CSS3 动画(animation)或 JS 结合使用,从而实现一些动态效果。



由于其存在较严重的兼容问题,IE 浏览器不支持、微软推出的新浏览器 Edge12+(斯巴达)支持,使得效果没有办法在所有的浏览器当中保持统一,因此,三维变形在网页开发当中使用相对较少,特别是在 PC 平台。

### 16.2.7 关于变形的问题区

(1) rotateX()中的 X 是否一定要大写?

可以不大写,但是建议大写。主要为了方便开发人员查看。

(2) 对于 rotateX(0deg),可否简写成 rotateX(0)?

原则上来说是可以简写的,但是在火狐的部分版本当中,无法正常读取无单位的角度值。因此,基于兼容性的考虑,建议不要省略单位。



## 16.3 CSS3 过渡

### 16.3.1 过渡的基本属性

一个视觉效果的变化可以是瞬间的,也可以是逐渐完成的。过渡就是让这种变化逐渐完成。

CSS3 过渡(transition)效果主要包括 4 个属性值,分别用于检索或设置对象中参与过渡的属性、对象过渡的时间、过渡的动画类型以及延迟对象过渡的时间。IE10+以及各个主流浏览器均支持。

#### 1. transition-property

transition-property 表示具体要“过渡”的属性,可以针对所有的属性进行应用,也可以只指定某一个属性或某几个属性应用过渡效果。

如果是针对所有属性,直接使用 all 即可。如果针对指定属性,则书写属性的名称。

#### 2. transition-duration

transition-duration 表示过渡所需要的时间,单位为 s(秒),这个时间指的是运动的总时间。

#### 3. transition-timing-function

transition-timing-function 表示过渡的方式。对于属性过渡的动画是有不同的过渡方法,不同过渡方法,主要体现在从起始到结束的过程中,不同阶段有着不同的运动速度。如下 5 种是常用的一些控制过渡样式的方式。

(1) linear: 表示线性过渡。

(2) ease: 表示平滑过渡。

(3) ease-in: 表示由慢到快的变化。

(4) ease-out: 表示由快到慢的变化。

(5) ease-in-out: 表示由慢到快再到慢的变化。

#### 4. transition-delay

transition-delay 表示过渡的延迟时间,单位为 s(秒)。

### 16.3.2 过渡的合写方法 transition

过渡有拆写与合写两种,拆写方式指的是针对 transition 的 4 种属性分别进行书写。合写,则是通过 transition 的属性进行书写,对于不同的属性过渡,使用逗号进行分隔。

基本语法(transition 中具体属性的书写顺序):

```
transition:[< transition - property > || < transition - duration > || < transition - timing -  
function > || < transition - delay >] [, [< transition - property > || < transition - duration >  
|| < transition - timing - function > || < transition - delay >]] *
```

代码实例:

```
transition: width 2s linear 1s;
```

代码解析:

该段代码表示:宽度发生过渡,过渡时长为 2s,延迟 1s 发生,过渡方式为“线性过渡”。

方法选用:

在开发时,推荐使用合写的 transition 属性,既能够减少代码量,又易于开发与维护。

### 16.3.3 多属性过渡时,各个属性的书写方法

功能需求:

当光标移入元素时:

边框颜色,从红色变成黑色,过渡时间为 0.5s,延时 0.1s,变化方式为“由慢到快”;

背景颜色,从浅蓝色变成浅灰色,过渡时间为 1.5s,延时 0.2s,变化方式为“线性”;

文字颜色,从黑色变成蓝色,过渡时间为 2.5s,延时 0.5s,变化方式为“平滑过渡”。

#### 1. 分写方法

代码实例:

```
<!doctype html >  
<html >  
<head >  
  <meta charset = "UTF - 8">  
  <title>HTML5 布局之路 - transition 拆写</title>  
  <link rel = "stylesheet" href = "../css/reset.css">  
  <style>  
    .box {  
      width: 150px;  
      height: 150px;  
      border: 5px solid red;  
      background: #39f;  
      transition - property: border - color, background - color, color;  
      transition - duration: .5s, 1.5s, 2.5s;  
      transition - timing - function: ease - in, linear, ease;  
      transition - delay: .1s, .2s, .5s;
```



```

    }
    .box:hover {
        color: blue;
        border: 5px solid black;
        background: #ccc;
    }
</style>
</head>
<body>
    <div class = "box"> HTML5 布局之路</div>
</body>
</html>

```

## 2. 合写方法(推荐)

代码实例:

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - transition 属性控制(合写)</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        .box {
            width: 150px;
            height: 150px;
            border: 5px solid red;
            background: #39f;
            transition: border .5s ease-in .1s,
                        background-color 1.5s linear .2s,
                        color 2.5s ease .5s;
        }
        .box:hover {
            color: blue;
            border: 5px solid black;
            background: #ccc;
        }
    </style>
</head>
<body>
    <div class = "box"> HTML5 布局之路</div>
</body>
</html>

```

### 16.3.4 过渡得以实现的必备要素

transition 指的是从一种状态到另一种状态,因此,需要拥有两种属性状态。通常情况下,一种默认状态是存在于 CSS 当中的,而另一种状态以 hover 的方式进行书写或通过 JS

进行动态变化。这种“过渡”的过程并不能够自行运行,需要开发工程师通过JS或浏览者与页面交互来实现。

### 16.3.5 关于过渡的问题区

(1) 过渡时长或延迟时间,小数点前面的零可以省略?

是的。如果过渡时长或过渡延迟时间不足1s,小数点前的0可以省略。以下两行代码等价。

```
transition: all 0.5s;  
transition: all .5s;
```

(2) 有没有其他的过渡方式?

有。除了常见的“ease”、“linear”等5种过渡方式之外,还有贝塞尔曲线(大学高等数学的知识),根据贝塞尔曲线的值绘制出的图像表示的是过渡过程中速度的变化情况。

基本语法:

```
cubic-bezier(number, number, number, number);
```

语法解析:

特定的贝塞尔曲线类型,4个数值需在 $[0, 1]$ 区间内。

常见的5种过渡方式也可以转换成贝塞尔曲线的数值。

linear: 线性过渡。等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)。

ease: 平滑过渡。等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)。

ease-in: 由慢到快。等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)。

ease-out: 由快到慢。等同于贝塞尔曲线(0, 0, 0.58, 1.0)。

ease-in-out: 由慢到快再到慢。等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)。

5种常见过渡方式对应的贝塞尔曲线图如图16.8所示。

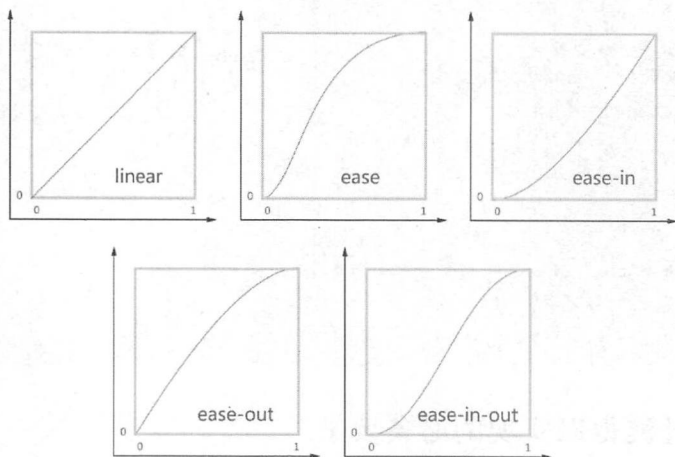


图 16.8 贝塞尔曲线图





## 16.4 CSS3 动画

### 16.4.1 帧与关键帧

帧：就是动画中最小单位的单幅影像画面，相当于电影胶片上的每一格镜头。在动画软件的时间轴上帧表现为一格或一个标记。

关键帧：相当于二维动画中的原画。指角色或者物体运动或变化中的关键动作所处的那一帧。关键帧与关键帧之间的每幅图像可以由软件来创建，这些处于关键帧之间的帧(图像)被称为过渡帧或者中间帧。

### 16.4.2 CSS3 动画的基本语法

#### 1. 定义动画

基本语法：

```
@-webkit-keyframes name {  
    0% {background-color: red;}  
    50% {background-color: blue;}  
    100% {background-color: white;}  
}  
  
@-webkit-keyframes move {  
    from {left: 0px; top: 0px;}  
    to {left: 400px; top: 400px;}  
}
```

语法解析：

关键帧由“@keyframes”开头，后面紧跟“动画名称”和一对花括号{}。在@keyframes与动画名称之间使用空格分隔。

@keyframes 表示定义关键帧，后面的动画名称表示的是这个动画的名字，在代码当中的 keyframes 前面添加了-webkit 前缀，表示谷歌浏览器(不同内核浏览器有着不同的写法)；

标签使用动画时需要通过这个名称寻找；

花括号中就是一些不同时间段的样式规则。

“@keyframes”中的样式由多个百分比构成，在0%~100%中间，可以创建多个百分比。这个百分比即在动画过程中的关键帧位置。也可以使用 from...to 来代表一个动画的开始和结束帧位置。

每一个百分比中都可以为动画效果元素加上不同的属性，从而让元素在一种不断变化的状态。

**注意：**当使用百分比表示起始帧时，使用“0%”，不能去掉百分号。keyframes 只接受百分比数值。

### 代码解析:

第一段代码的含义是: 创建一个名为“name”的 CSS3 动画, 这个动画在 0% 时元素的背景颜色为红色, 50% 时元素的背景颜色为蓝色, 100% 时元素的背景颜色为白色。在 0% ~ 50%、50% ~ 100% 之间由计算机自动计算并变化。

第二段代码的含义是: 创建一个名为“move”的 CSS3 动画, 这个动画在最初(0%)时 left 和 top 值均为 0, 在最终(100%)left 和 top 值均要达到 400px。

## 2. 使用动画

上面的一系列代码只是定义了动画, 接下来再具体地应用到动画元素当中, 通过 animation 属性来使用这个动画, 才能够实现想要的效果。

接下来看一下 animation 动画的基本属性。

### 16.4.3 animation 动画的基本属性

CSS3 动画(animation)主要包括 6 个属性值, 分别是动画属性名、动画持续的时间、动画的频率、动画延迟时间、动画循环次数、动画方式。IE10+ 以及各个主流浏览器均支持, 移动端 Android 4 以前的浏览器不支持。

#### 1. animation-name

animation-name 表示“动画名称”, 也就是前面 keyframes 定义的动画名。通过该属性, 将元素与定义的动画连接到一起。

#### 2. animation-duration

animation-duration 表示动画所需要的时间, 单位为 s(秒), 这个时间指的是运动的总时间。

#### 3. animation-timing-function

animation-timing-function 表示动画频率, 与过渡方式类似, 拥有 5 种常用方式, 也可以直接设置贝塞尔曲线值。

(1) linear: 表示线性过渡。

(2) ease: 表示平滑过渡。

(3) ease-in: 表示由慢到快的变化。

(4) ease-out: 表示由快到慢的变化。

(5) ease-in-out: 表示由慢到快再到慢的变化。

(6) cubic-Bezier (<number>, <number>, <number>, <number>): 通过设置贝塞尔曲线值来控制变化速度。

#### 4. animation-delay

animation-delay 表示动画的延迟时间, 单位为 s(秒)。

#### 5. animation-iteration-count

animation-iteration-count 表示动画的循环次数, 默认为不循环, 设置数字即可, 如果希望无限次循环, 可以将其属性值设置为“infinite”。



## 6. animation-direction

animation-direction 表示动画的方式,或者说是动画播放的方向。只存在两个属性值,默认值为 normal,如果设置为 normal 时,动画的每次循环都是向前播放;另一个值是 alternate,动画播放在第奇数次向前播放,第偶数次向反方向播放。

## 7. 其他的属性

CSS3 动画还提供了 animation-fill-mode 等属性,在实际开发当中使用相对不多,在此就不再详细介绍了。

### 16.4.4 动画命令的合写方法 animation

动画和过渡一样,也有拆写与合写两种,拆写方式指的是针对 animation 的各种属性分别进行书写。合写,则是通过 animation 属性进行书写。多个动画之间使用逗号分隔。

基本语法(animation 中具体属性的书写顺序):

```
animation:[<animation-name> || <animation-duration> || <animation-timing-function> ||  
<animation-delay> || <animation-iteration-count> || <animation-direction>]  
[, [<animation-name> || <animation-duration> || <animation-timing-function> || <animation-  
delay> || <animation-iteration-count> || <animation-direction>] ] *
```

代码实例:

```
.all { -webkit-animation: move 5s linear infinite; }
```

代码解析:

该段代码表示:应用名称为“move”的动画,动画总时长为 5s,不延迟,线性过渡,无限循环。

方法选用:

在开发时,推荐使用合写的 animation 属性,既能够减少代码量,又易于开发与维护。

### 16.4.5 动画与过渡的比较

(1) transition 只能指定起始状态(from)和结束状态(to),animation 可以有多个关键帧设置。

(2) transition 需要有动作(hover 等)来触发才能执行,animation 可以自己执行。

(3) 两者支持的属性不同,animation 多出 iteration-count, direction, play-state 等属性,可控性更强。

(4) transition 的作用是“平滑改变 CSS 值”,animation 的作用是创建“动画效果”。

# 第17章

## 各章节自评习题集

针对第1章到第16章,出于加强理论知识以及基础性知识的考虑,设置了相对应的习题集。

建议在进行案例实战开发之前,先排查自己理论和基础代码方面的漏洞,只有扎实掌握知识,才能够更好更灵活地应用。

习题集主要用于考查理论类知识和比较基础小型的实战类知识,对于更多的实战练习,请看第18章。另外,出于每个习题集篇幅的考虑,将第8、9章合并为一个习题集;将第10、11章合并为一个习题集。

在习题的解析当中,针对习题进行了分类,主要有4种标识,分别是【记忆】、【理解】、【应用】、【考】。在完成习题测试之后,可通过这几种标识来分析自己当前的学习漏洞或提升自己知识的系统化程度。

**【记忆】:** 如果记忆类知识错误较多,说明基础不够扎实牢固,在这种情况下,会对一些知识的理解造成限制,也会对开发速度、开发质量造成一定的影响。建议在查看书籍的时候,注重一些细节,这样能够让记忆变得更深刻;另外,也需要拿出专门的时间记忆知识。

**【理解】:** 如果理解类知识错误较多,说明在一些知识的掌握上有可能停留在表面,掌握的深度不够,知其然而不知所以然。建议遇到一些内容时,不仅是看,更重要的是多思考,思考的方法可以从“(知识点)是什么,为什么(为什么会用这个知识、为什么会有这种现象等),怎么做(知识如何应用于实践当中)”出发。

**【应用】:** 如果应用类知识错误较多,说明在看书学习时,很有可能更多地关注于理论知识,而忽略了实践能力。有些知识和经验是通过读书得来的,而对于开发、写代码来说,代码实战是知识和经验的“直接来源”,也是不可或缺的一部分。建议多加强练习,例如,针对各个属性,书写实例,查看效果,增强理解;对于在学习过程中出现的疑惑,通过代码实战来验证自己的猜想或假设;自我主动思考,实现一些书上没有提到的案例效果等。

**【考】:** 标注“考”的问题,指的是在开发工程师技术面试过程中,比较常见的面试题,需要找工作的朋友可以通过此部分辅助自己的复习和准备,不找工作的朋友可以通过此部分“抓取”一些重点。

声明:虽然针对每章设置了习题,但是这些习题并不能够涉及所有的知识细节,因此该习题集只是辅助进行复习。





## 17.1 习题集 01

### 17.1.1 习题内容

对应章节：第1章

#### 旅途之前

涉及的主要知识：

- 讲解方法与旅途指南；
- 什么是HTML5；
- HTML5行业以及职业；
- Photoshop的使用；
- 编辑器的使用与浏览器调试。

#### 一、单项选择题

- 针对本书的讲法，描述不正确的是( )。
  - 理论知识与实战练习相结合
  - 按照网页开发流程进行知识的讲解
  - 由浅入深，循序渐进，注重理论与实践相结合
  - 字典式的知识罗列
- 对于 Photoshop 软件的使用，如下选项当中不属于 HTML5(前端)开发工程师职责的是( )。
  - 设计网页图像
  - 针对设计图进行图片切图
  - 处理切好的图片并将图片保存成合适的格式
  - 背景图合并处理

#### 二、判断题

- 开发页面时应当及时调试，且测试多种浏览器，以防止太久之后发现问题无法解决。
- 编写 HTML5 代码，必须使用推荐的 Sublime、H5Builder、Dreamweaver 等专业开发工具，其他的文件不能够进行开发。
- HTML5 能够独立完成苹果或安卓 APP 的开发。

#### 三、填空题

- HTML5 的基本组成是\_\_\_\_\_ + \_\_\_\_\_ + \_\_\_\_\_。
- 使用 rgb 数值表示(纯)红色：\_\_\_\_\_。
- 浏览器调试时，调试方法与工具包括：\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。

#### 四、排序题

本书推荐的学习流程是\_\_\_\_\_。

- 对应章节自评测试题
- 在适当章节之后进行较大型(涉及多章节的)案例实战
- 学习基本技术知识
- 对应章节代码实战练习

## 五、问答题

1. 什么是 HTML5?
2. 什么是“W3C”?
3. 解释“页面重构”。
4. 简述 JPG、GIF、PNG 三者的区别。

### 17.1.2 习题答案

#### 一、单项选择题

1. 【理解】D
2. 【理解】A

#### 二、判断题

1. 【应用】对。
2. 【理解】错。(文本类编辑器均可,使用推荐的开发工具只是为了更加快捷方便)
3. 【记忆】错。(能和 Object C 或 Java 配合,实现混合应用程序的开发)

#### 三、填空题

1. 【考】【记忆】“HTML+ CSS + JavaScript”或“结构+样式+行为”
2. 【理解】rgb(255, 0, 0)
3. 【考】【记忆】谷歌火狐等浏览器的控制台(F12)、IE 的 debug、火狐的 FireBug 工具

#### 四、排序题

【应用】C A D B

#### 五、问答题

1. 【考】【应用】答题思路:可以从广义角度和狭义角度分别剖析 HTML5,并描述关于 HTML5 的历史。请查阅第 1 章的具体知识。
2. 【考】【记忆】略。请查阅第 1 章的具体知识。
3. 【考】【记忆】略。请查阅第 1 章的具体知识。
4. 【考】【理解】略。请查阅第 1 章的具体知识。



## 17.2 习题集 02

### 17.2.1 习题内容

对应章节:第2章

#### HTML5入门

涉及的主要知识:

























- 网站开发流程;
- HTML文件的基本结构;
- HTML书写规则;
- HTML头文件(文档声明、元信息、标题等);
- HTML常用标签介绍。

#### 一、单项选择题

1. 如下网站流程中的具体步骤,哪种不属于前端开发工程师的工作职责?( )





- A. 与设计师对接设计图                      B. 与运维人员共同推广  
C. 与后台共同完成网站开发                D. 与产品经理对接需求
2. 如下开发原则当中,最好(也是所推荐)的一种方式( )。
- A. 做好责任规避,百分百还原设计的图,出了问题设计的责任  
B. 设计图本身会有一定误差,前端在开发过程中自行调整即可  
C. 设计图本身会有一定误差,前端在开发过程中根据具体情况调整,但误差不大于 3  
D. 设计图本身会有一定误差,对于有歧义的地方与设计沟通,无歧义的地方做到百分百还原设计图
3. 在 HTML 代码中使用注释的作用不包括( )。
- A. 让代码变得更漂亮                      B. 增强代码的可阅读性  
C. 对于多人合作项目增强配合度        D. 辅助进行布局的错误调试
4. 如下文件夹结构以及文件命名中,最差的方案是( )。
- A.  project 项目文件夹
-  cn      存放中文网页文件
  -  en      存放英文网页文件
  -  images    存放图像文件
  -  css      存放CSS样式表
  -  js      存放JS脚本
- B.  project 项目文件夹
-  index.html 网页首页
  -  cn      存放所有网页文件
  -  images    存放图像文件
  -  css      存放CSS样式表
  -  js      存放JS脚本
- C.  project 项目文件夹
-  首页.html 网页首页
  -  网页      存放所有网页文件
  -  图像      存放图像文件
  -  样式表    存放CSS样式表
  -  脚本      存放JS脚本
- D.  project 项目文件夹
-  index.html 网页首页
  - .....
  -  网站的所有的网页 (html) 文件
  -  images    存放图像文件
  -  css      存放CSS样式表
  -  js      存放JS脚本

## 二、判断题

- 只要在网页(HTML)文件中的< head ></head>标签当中,最先书写< meta charset = “utf-8”>,就可以避免网页在浏览器中的乱码问题。
- 在书写标签时,使用大写字母(如: <! DOCTYPE HTML >),浏览器并不会错误解析。
- 在各个标签当中,属性值两侧的引号可以使用中文或英文的单引号或双引号,重点是自我保持一致即可。
- 为了让网页标题能够正常显示,会将< title >放置于“字符编码设置”之后。
- 使用 Tab 键或空格进行缩进,都可以被浏览器正常解读,但最好自我保持一致。

## 三、填空题

- HTML5 的基本语言组成是\_\_\_\_\_ + \_\_\_\_\_ + \_\_\_\_\_(此处三个空填写英文全拼),也就是人们平时所说的一个网页中的\_\_\_\_\_,\_\_\_\_\_和\_\_\_\_\_。



2. 网页文档的加载是自\_\_\_\_\_而\_\_\_\_\_进行的。

3. 通常一个网站的首页命名为\_\_\_\_\_或\_\_\_\_\_。

4. 请罗列至少 15 种 HTML 的基本标签: \_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、  
\_\_\_\_\_, \_\_\_\_\_、\_\_\_\_\_, \_\_\_\_\_、\_\_\_\_\_, \_\_\_\_\_、\_\_\_\_\_, \_\_\_\_\_、\_\_\_\_\_,  
\_\_\_\_\_, \_\_\_\_\_、\_\_\_\_\_。

5. 在 HTML 标签当中,用于布局的是\_\_\_\_\_标签。

#### 四、基础类代码题

1. 根据需求实现代码书写

```
<_____> 声明文档为一个 HTML 文档
<_____> 包含整个 HTML 文档的标签
<_____> HTML 文档的头部
  <meta_____> 设置字符编码为 utf-8
  <_____> </_____> 设置网页标题为: 我的 HTML5 布局之路
  <_____> > 设置网页描述信息为: 第一个知识自评文件
  <_____> > 设置网页关键字为: HTML5 布局
  <_____> > 设置网页作者为: 你的大名 - !
</_____> 关闭 HTML 文档的头部
<_____> HTML 文档的内容区
  <div>开启 HTML5 旅途</div>
</_____> 关闭 HTML 文档的内容区
</_____> 关闭 HTML 文档
```

2. 如何实现页面 10s 后刷新到 <http://www.h5course.com> 的网址?

### 17.2.2 习题答案

#### 一、单项选择题

1. 【理解】B
2. 【应用】D
3. 【理解】A
4. 【记忆】C

#### 二、判断题

1. 【理解】错。(文件内部字符编码、文件编码、浏览器编码均保持一致时才能避免乱码问题)
2. 【记忆】对。
3. 【记忆】错。(不允许使用中文符号)
4. 【理解】对。
5. 【应用】对。

#### 三、填空题

1. 【考】【记忆】HTML、CSS、JavaScript; 结构、样式、行为。
2. 【理解】自“上”而“下”。





3. 【记忆】index、default。
4. 【记忆】略。请查阅第2章的具体知识
5. 【记忆】div。

#### 四、基础类代码题

##### 1. 【应用】

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>我的 HTML5 布局之路</title>
  <meta name = "description"content = "第一个知识自评文件">
  <meta name = "keywords"content = "HTML5, 布局">
  <meta name = "author" content = "刘国利">
</head>
<body>
  <div>开启 HTML5 旅途</div>
</body>
</html>
```

##### 2. 【考】【应用】

```
<meta http - equiv = "refresh" content = "10; url = http://www.h5course.com">
```



## 17.3 习题集 03

### 17.3.1 习题内容

对应章节：第3章

#### 整体布局（上） —— 标签尺寸处理

涉及的主要知识：

- CSS基本概念；
- 三种不同的CSS引入方式；
- CSS基本选择器（标签名、类名、id名）；
- CSS编码规范；
- 样式重置；
- 盒模型。

#### 一、单项选择题

1. 使用外部引入的CSS引入方式的意义不包括（ ）。
  - A. 增强代码的可维护性
  - B. 降低冗余代码比例
  - C. 防止服务器请求压力
  - D. 加快网站中首页的加载速度
2. 掌握CSS选择器优先级的“根本”目的在于（ ）。
  - A. 在审查代码错误时能够迅速找到问题所在
  - B. 能够清晰地说出选择器的级别关系
  - C. 防止出现不必要的样式覆盖，并在部分情况下利用优先级的覆盖，实现一些特殊效果



- D. 应对面试
3. 如下关于盒模型的理解错误的是( )。
- A. 每个盒模型都必须设置 5 种基本属性
  - B. 可以把 body 中的每个元素都理解成一个盒子
  - C. 可以把 body 理解成一个盒子
  - D. 对于每个盒模型,默认的表现样式并不完全相同
4. 对于“使用 background,而非 border 属性来标示元素”的描述有误的是( )。
- A. 不占用空间大小,后期删除属性并不会引发盒模型大小变化
  - B. 通常在布局时使用,主要目的在于令每个盒模型可视化
  - C. 将元素的背景样式单独设置会比放置在原有的选择器中书写更好
  - D. 主要是为了视觉上布局时更漂亮

## 二、判断题

1. CSS 外部引入方式必然会导致服务器请求压力,所以要慎重使用。
2. 标签选择器不能随便使用的主要原因在于选择范围过广。
3. CSS 代码书写顺序需要遵循显示样式-自身样式-文本样式的原因只是开发者觉得那么看着效果比较好。
4. div 元素默认独自占据一行空间,其中,一行指的是当前窗口的最大宽度。
5. link 引入的 CSS 文件以及 style 标签中书写的样式,可以放置在 body 元素当中。

## 三、填空题

1. CSS 基本选择器包括\_\_\_\_、\_\_\_\_、\_\_\_\_三种。
2. 行内样式与基本选择器的优先级级别关系是: \_\_\_\_\_ > \_\_\_\_\_ > \_\_\_\_\_ > \_\_\_\_\_。
3. 可以使用\_\_\_\_\_标签,将 CSS 文件引入到网页当中,而该标签的 rel 属性用于定义\_\_\_\_\_。
4. 盒模型属性包括\_\_\_\_、\_\_\_\_、\_\_\_\_、\_\_\_\_、\_\_\_\_\_。
5. 常见的三种边框样式是: \_\_\_\_\_、\_\_\_\_、\_\_\_\_(书写单词和含义)。

## 四、问答题

1. 请描述“页面回流”与“页面重绘”。
2. 简述 border: 0;与 border: none;的区别。
3. 请描述“不同 CSS 引入方式的优劣势”。

## 五、基础类代码题

1. 代码样式与结构:

```
<style>
    .box1 {
        margin: 30px;
    }
    .box2 {
        margin: 20px;
```





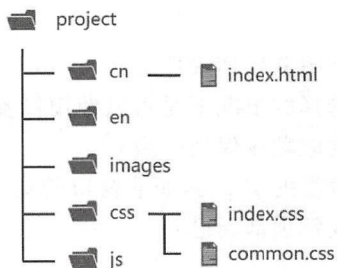
```

    }
</style>
<div class = "box1"></div>
<div class = "box2"></div>

```

两个 div 之间的距离是多少像素?

2. 请查看如下文件的关系,之后,书写“HTML 文件”当中引入“common.css”使用的路径。



3. 请查看如下代码,之后描述出第二个 div 的样式。

```

<style>
    div {
        height: 200px;
        border: 1px solid red;
    }
    .box {
        margin: 10px auto;
    }
    #box {
        padding: 20px;
        border: 10px solid black;
    }
</style>
<div class = "box">第一个 div</div>
<div id = "box" class = "box">第二个 div</div>
<div id = "box">第三个 div</div>
<div>第四个 div</div>

```

4. 请根据如下需求书写样式代码。

需求: 为页面中所有的 div 元素设置盒模型属性,宽度 400px、高度 200px,上外边距 20px、下外边距 10px,左右居中(在父级元素中),上下内边距 10px、左右内边距为 12px。要求使用尽可能简单的方式进行书写。

5. 请根据如下需求书写样式代码。

需求: 为类名为 wrap 的元素设置黑色实线边框,上边框 20px,左边框是 10px,其他边框都为 4px。要求使用尽可能简单的方式进行书写。



### 17.3.2 习题答案

#### 一、单项选择题

1. 【理解】C
2. 【应用】C
3. 【记忆】A
4. 【理解】D

#### 二、判断题

1. 【理解】错。(有可能产生服务器请求压力)
2. 【理解】对。(合理分析选择的情况下是可以使用标签名选择器的)
3. 【记忆】错。(和浏览器读取顺序保持一致)
4. 【理解】错。(占据父级宽度的 100%，而非窗口的宽度)
5. 【记忆】对。(只是推荐放到顶部而已)

#### 三、填空题

1. 【记忆】id 选择器、标签名选择器、类名选择器
2. 【记忆】行内样式> id 选择器>类名选择器>标签名选择器
3. 【记忆】link、外部文件的类型(或外部文件的关联类型)
4. 【记忆】外边距(margin)、内边距(padding)、边框(border)、宽度(width)、高度(height)
5. 【记忆】solid(实线)、dotted(点线)、dashed(虚线)

#### 四、问答题

1. 【考】【记忆】略。请查阅第 3 章的具体知识。
2. 【考】【记忆】略。请查阅第 3 章的具体知识。
3. 【考】【理解】略。请查阅第 3 章的具体知识,注意不要漏掉@import。

#### 五、基础类代码题

##### 1. 【考】【应用】

30px。默认状态下纵向 margin(外边距)叠加。

##### 2. 【应用】

路径: ../css/common.css

##### 3. 【应用】

宽度默认,高度 200px,10px 黑色实线边框、20px 的内边距值(4 个方向)、10px 的纵向外边距值、水平方向外边距自动,但由于没有设置宽度,因此并没有“居中”的视觉效果。

##### 4. 【应用】

```
div {  
    width: 400px;  
    height: 200px;  
    margin: 20px auto 10px;  
    padding: 10px 12px;  
}
```





## 5. 【应用】

```
.wrap {  
    border: 4px solid black;  
    border - width: 20px 4px 4px 10px;  
}
```



## 17.4 习题集 04

对应章节：第4章

涉及的主要知识：

## 整体布局（下）

## —— 浮动布局

✿ 浮动

✿ 清除浮动

## 17.4.1 习题内容

## 一、单项选择题

1. 为一个元素设置浮动,( )。
  - A. 一定会对设定浮动的元素自身产生影响
  - B. 一定会导致浮动元素的父级高度产生变化
  - C. 一定会使浮动元素父级的兄弟级元素布局发生变化
  - D. 一定会对浮动元素的兄弟级元素的位置产生影响
2. 在浮动元素的父级元素中进行伪元素清浮动的操作,不能够( )。
  - A. 防止浮动元素的兄弟级受到影响
  - B. 让父级的高度恢复正常的内容撑开
  - C. 让浮动元素不再浮动
  - D. 防止父级元素的兄弟级受到影响
3. 利用伪元素清浮动的好处不包括( )。
  - A. 避免使用过多的空标签
  - B. 提高代码的可读性
  - C. 可以避免兼容问题
  - D. 使用起来比较简单

## 二、判断题

1. 清浮动就是让父级恢复正常高度。
2. 空标签清浮动与伪元素清浮动的原理基本相同。
3. 伪元素清浮动必须使用 clearfix 的类名。
4. 在开发当中,可以根据实际情况,嵌套多层标签,让浮动布局更灵活。

## 三、填空题

1. IE6、IE7 不兼容 clear: both;需要为其添加\_\_\_\_\_命令。
2. br 清浮动主要是通过将它\_\_\_\_\_属性设置为\_\_\_\_\_。

## 四、问答题

请详细阐述对“浮动”和“清除浮动”的理解。(提示:该题为大型问答题,需要有逻辑地



阐述出所了解的知识。)

## 五、基础类代码题

1. 实现一个布局,分为左中右三栏(这三个元素的父元素宽度始终大于 300px)。左栏固定宽为 100px,右栏固定 200px,中栏随屏幕宽自动适应。(提示:浮动、margin 值的结合应用。)

2. 请实现如图 17.1 所示效果。

(1) 假设没有给定结构,应当如何实现?

(2) 假设给定结构,要求 div 顺序不能更换,应当如何实现?(提示:浮动、清除浮动、margin 值的结合应用。)

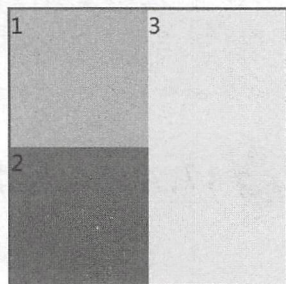


图 17.1 效果图

## 17.4.2 习题答案

### 一、单项选择题

1. 【理解】A

2. 【理解】C

3. 【理解】C

### 二、判断题

1. 【记忆】错。(除了解决父级塌陷的问题之外,还有防止该元素之后的兄弟级元素布局受到影响)

2. 【理解】对。

3. 【记忆】错。(类名可以随意设置,但请考虑语义性)

4. 【应用】对。(请在实战时多思考,灵活应用所学知识)

### 三、填空题

1. 【记忆】float: none;

2. 【考】【记忆】clear, all

### 四、问答题

【考】【理解】

答案提示,建议可以按照如下思路整理答案。

在最初 HTML 刚刚发明时,并没有让布局以及大部分的元素既能够 (1),又能够 (2)。CSS 中的浮动,主要是为了解决上面的这个问题。设置浮动之后,就能够更方便地进行网页布局了。

但是,浮动会导致元素 (3)。当元素 (3) 之后,并不会占据 (4),会导致父级元素 (5),也会对浮动元素之后的其他 (6) 级元素造成布局影响。

为了防止这两类影响,就需要及时清除浮动。清除浮动包括两类,一类是清除掉对 (6) 级元素的影响,只需要为相应元素设置 (7) 属性,就可以直接清除掉浮动元素对该元素自身的影响;另一类是清除掉对父级元素的影响,主要包括 (8) 等常见方法。其中使用较为频繁的是 (9),该方法与 (10) 方法原理相同,其实现的基本代码为





(11)       , 代码含义为 (12)       。其他方法各自有其优劣势 (13)       。

- (1) 设置宽高
- (2) 与其他元素处于同一行
- (3) 脱离文档流
- (4) (物理)空间
- (5) 高度塌陷
- (6) 兄弟
- (7) clear: both/left/right;
- (8) 空标签清除浮动、br 标签清除浮动、overflow: hidden、overflow: auto、父级元素浮动、after 伪元素清除浮动
- (9) after 伪元素清除浮动
- (10) 空标签清除浮动
- (11) {content: "\200B"; clear: both; display: block; height: 0;}
- (12) 略, 请查阅第 4 章的具体知识。
- (13) 此处罗列即可。具体内容略, 请查阅第 4 章的具体知识。

## 五、基础类代码题

### 1. 【考】【应用】

代码实例:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 三栏布局</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      min-width: 300px;
      /* 该代码用于实现"假设父级宽度始终大于 300px"的需求,min-width 表示最小宽度,在第 9 章会详细讲解 */
      border: 1px solid red;
    }
    .left {
      float: left;
      width: 100px;
      height: 100px;
      background: #ccc;
    }
    .right {
      float: right;
      width: 200px;
      height: 100px;
      background: #f0f0f0;
```

```

    }
    .middle {
        margin: 0 200px 0 100px;
        height: 100px;
        background: #39f;
    }
</style>
</head>
<body>
    <div class = "wrap">
        <div class = "left"></div>
        <div class = "right"></div>
        <div class = "middle"></div>
    </div>
</body>
</html>

```

#### 代码解析：

左边和右边的块分别左右浮动，此时这两个元素脱离文档流而存在，只剩下第三个 div（也就是效果表现上的中间那个块）。

第三个 div 会按照父级元素中没有 left 和 right 的状态进行排布，但是此时该 div 的左侧和右侧内容会与 left 和 right 块的内容重叠，因此为其设置 margin 值。由于 width 没有设置固定值，会随着浏览器窗口的大小变化而发生变化。

父级元素中的“min-width: 300px;”一句代码，用于实现“假设父级宽度始终大于 300px”的需求，如果去掉这个代码，当浏览器缩小到 300px 以下时，布局会错乱。min-width 表示最小宽度，在第 9 章会详细讲解。

**备注：**请务必注意三个 div 的书写顺序，这个顺序至关重要。

#### 2. 【考】【应用】

(1) 未给定结构时的实现方法。

使用结构（比较容易实现的结构）：

```

<div class = "wrap clearfix">
    <div class = "first">1 </div>
    <div class = "third">3 </div>
    <div class = "second">2 </div>
</div>

```

#### 代码实例：

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - 特殊浮动布局</title>
    <link rel = "stylesheet" href = "../css/reset.css">

```



```
<style>
    .wrap {
        width: 200px;
        border: 2px solid red;
    }
    .first {
        float: left;
        width: 100px;
        height: 100px;
        background: #ccc;
    }
    .third {
        float: right;
        width: 100px;
        height: 200px;
        background: #f0f0f0;
    }
    .second {
        float: left;
        width: 100px;
        height: 100px;
        background: #39f;
    }
</style>
</head>
<body>
    <div class = "wrap clearfix">
        <div class = "first"> 1 </div>
        <div class = "third"> 3 </div>
        <div class = "second"> 2 </div>
    </div>
</body>
</html>
```

#### 代码解析:

第一个元素左浮动,第二个(内容为3)元素右浮动,第三个(内容为2)元素左浮动。由于元素浮动方向不同,因此在浮动过程中并不互相影响。

按照浏览器自上而下加载的模式,此处的元素排布顺序至关重要。

(2) 给定结构的实现方法。

#### 给定结构:

```
<div class = "wrap clearfix">
    <div class = "first"> 1 </div>
    <div class = "second"> 2 </div>
    <div class = "third"> 3 </div>
</div>
```

## 代码实例：

```
<!doctype html >
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 无给定结构的特殊浮动布局</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 200px;
      border: 2px solid red;
    }
    .first {
      float: left;
      width: 100px;
      height: 100px;
      background: #ccc;
    }
    .second {
      clear: left;
      float: left;
      width: 100px;
      height: 100px;
      background: #39f;
    }
    .third {
      float: right;
      width: 100px;
      height: 200px;
      margin: -100px 0 0;
      background: #f0f0f0;
    }
  </style>
</head>
<body>
  <div class = "wrap clearfix">
    <div class = "first">1</div>
    <div class = "second">2</div>
    <div class = "third">3</div>
  </div>
</body>
</html>
```

## 代码解析：

为第二个 div 元素设置清除浮动+自身左浮动,这样第二个元素就不能够显示在第一个元素的右侧,而是从第一个元素的底部开始向左浮动;为第三个 div 元素设置右浮动,也将第一个 div 元素的底部开始向右浮动。此时为了让第三个元素能够移动到和第一个 div“持平”,为其设置 margin-top 的负值,属性值为-100px(第一个元素的高度)。





## 17.5 习题集 05

### 17.5.1 习题内容

对应章节：第5章

#### 模块布局（上）

#### ——选择标签

涉及的主要知识：

- ✱ HTML模块布局用的各类标签；
- ✱ 标签选择的影响因素；
- ✱ 标签默认显示样式与display；
- ✱ 标签嵌套规则；
- ✱ 搜索引擎优化——SEO；
- ✱ 后代、群组、子代等“加强版”选择器。

#### 一、单项选择题

1. 如下嵌套关系错误的是( )。

A. <pre>&lt;div&gt;   &lt;a&gt;     &lt;img /&gt;   &lt;/a&gt; &lt;/div&gt;</pre>	B. <pre>&lt;p&gt;   &lt;h2&gt;&lt;/h2&gt;   &lt;div&gt;&lt;/div&gt; &lt;/p&gt;</pre>	C. <pre>&lt;a&gt;   &lt;span&gt;     &lt;strong&gt;&lt;/strong&gt;     &lt;em&gt;&lt;/em&gt;   &lt;/span&gt; &lt;/a&gt;</pre>	D. <pre>&lt;ins&gt;   &lt;p&gt;&lt;/p&gt; &lt;/ins&gt;</pre>
--	---	--	---

2. li 标签的默认 display 值是( )。

- A. block                      B. inline-block                      C. list-item                      D. inline

3. 如下选择器当中,优先级最高的是( )。

- A. div #con { }
- B. #div con { }
- C. .con .a .b .c .d .e .f .g .h .i .j .k .l .m .n .o { }
- D. .con .test p { }

#### 二、判断题

- 并不推荐标签错误嵌套,但是,即便嵌套错误,浏览器也能够正常解读。
- dl 下只能出现 dt 和 dd,但是 dt 和 dd 当中可以嵌套别的标签。

#### 三、填空题

- HTML 标签可以简单地划分为\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_三大类。请分别为三类标签举例(每个类型 5 种标签以上)\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
- 标签选择的影响因素包括\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
- SEM(搜索引擎营销) = \_\_\_\_\_ + \_\_\_\_\_。
- 在百度等搜索引擎的搜索列表当中,介绍性的信息来自\_\_\_\_\_。
- 目前学过的 CSS 选择器包括\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。

#### 四、问答题

1. 简述锚文本的含义。
2. 简述块元素与行元素的区别。
3. 简述 display: none 与 visibility: hidden 的区别。
4. 简述标签的嵌套规则。
5. 简述 strong、em、b、i 的区别。

#### 五、基础类代码题

1. 书写出如下 indexNum 编号 1~3 的 div 的样式。

代码：(左结构,右样式)

```
<div class = "wrap" indexNum = "1">
  <div class = "con" indexNum = "2"></div>
  <div class = "con">
    <div indexNum = "3"></div>
  </div>
</div>
```

```
.wrap div.con {
  background: #39f;
}
.wrap div .con {
  border: 1px solid #f00;
}
.wrap .con {
  padding: 10px;
  background: #00f;
}
.wrap .con div {
  margin: 10px;
  background: #000;
}
```

2. 书写出如下 indexNum 编号 4~6 的 div 的样式。

代码：(左结构,右样式)

```
<div class = "wrap" indexNum = "4">
  <div class = "con"></div>
  <div class = "con test" indexNum = "5">
    <div indexNum = "6"></div>
  </div>
</div>
```

```
.wrap div {
  background: #39f;
}
.wrap, .con div, .test {
  border: 1px solid #f00;
}
.test div {
  padding: 10px;
  background: #00f;
}
```

### 17.5.2 习题答案

#### 一、单项选择题

1. 【记忆】B



2. 【记忆】C

3. 【理解】B

## 二、判断题

1. 【记忆】错。(有些情况会出现问题,请查阅第5章的具体知识)

2. 【记忆】对。

## 三、填空题

1. 【考】【记忆】块元素、行元素、第三类元素

块元素: `div p h1 h2 dl dt dd ul ol`; 行元素: `a img span strong em`; 第三类: `li table td tr th`。

备注: 很容易将 `li` 分错类。

2. 【理解】默认样式、嵌套层数、语义性(考虑 SEO)、扩展性(考虑标签复用性与后台)、样式可控性、特殊功能与需求

3. 【记忆】SEO(搜索引擎优化)、PPC(按照点击量付费)

4. 【记忆】元信息(meta)中的描述信息(description)

5. 【记忆】标签名选择器、id 选择器、类名选择器、后代选择器、群组选择器、子代选择器、通配符选择器、毗邻选择器

## 四、问答题

1. 【考】【记忆】略,请查阅第5章的具体知识

2. 【考】【理解】略,请查阅第5章的具体知识

3. 【考】【记忆】略,请查阅第5章的具体知识

4. 【考】【记忆】略,请查阅第5章的具体知识

5. 【考】【理解】略,请查阅第5章的具体知识

## 五、基础类代码题

### 1. 【应用】

编号1的div: 默认样式,没有通过CSS代码为编号为1的div设置样式。

编号2的div: `padding: 10px; background: #39f;`

编号3的div: `margin: 10px; background: #000;`

### 2. 【应用】

编号4的div: `border: 1px solid #f00;`

编号5的div: `border: 1px solid #f00; background: #39f;`

编号6的div: `border: 1px solid #f00; padding: 10px; background: #00f;`



## 17.6 习题集 06

对应章节: 第6章

### 模块布局(下)

#### —— 可用性与扩展性

涉及的主要知识:

- ◆ 超链接a标签;
- ◆ 鼠标不同状态的样式;
- ◆ 数据图与背景图;
- ◆ 高度控制与超出隐藏;
- ◆ overflow属性。

17.6.1 习题内容

一、单项选择题

1. 为列表页的描述性文章部分的 div, 设置了固定宽高, 最多显示三行文本内容, 当前后台传递过来文本数量略多(超过了 div 当前的大小), 如下操作方案中最为合理的是( )。
- A. 为 div 设置超出隐藏
- B. 为 div 设置单行超出显示为省略号
- C. 为 div 设置超出显示为滚动条, 以便查看其余的内容
- D. 为 div 设置 overflow: auto
2. 对于内容页的文章大标题, 假设该标题元素的类名为 tit, 标题中文字的数量不确定, 那么以下哪种设置最合理? ( )

A. .tit { overflow: hidden; height: 44px; line-height: 44px; }	B. .tit { overflow: auto; height: 44px; line-height: 44px; }
C. .tit { overflow: hidden; word-break: keep-all; white-space: nowrap; text-overflow: ellipsis; height: 44px; line-height: 44px; }	D. .tit { line-height: 44px; }

二、判断题

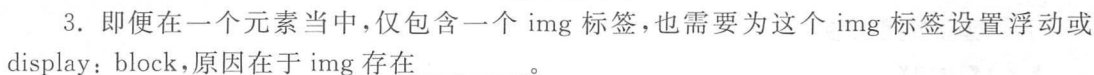
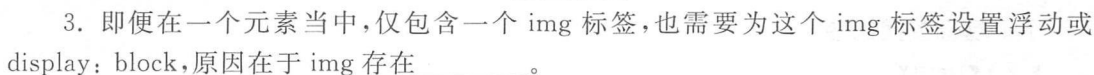
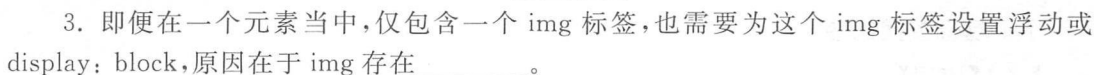
1. 为超链接设置 href="#", 单击超链接时, 页面不会发生任何变化。
2. 锚链接不仅可以调整到当前网页的某个位置, 还可以调整到其他页面的指定位置。
3. 有时需要为标题、段落内的 a 标签或 img 之外的 a 标签设置“display: block;”或“float: left/right”, 主要目的在于调整 a 标签的大小, 从而让用户可单击的区域变得更大, 提升用户体验。
4. 在超出的控制上, 通常单行文本“超出显示为省略号”, 多行文本“超出显示为隐藏”。
5. 在处理“N 条数据, N-1 条分隔线”的需求时, margin 负值的方法要比: first-child 好得多。

三、填空题

1. a 标签包含 4 个伪类选择器, 分别是(按照设置顺序书写): \_\_\_\_\_、  
\_\_\_\_\_, \_\_\_\_\_、\_\_\_\_\_。



2. 设置光标样式变成小手的命令是：\_\_\_\_\_。

3. 即便在一个元素当中，仅包含一个标签，也需要为这个标签设置浮动或display: block,原因在于存在\_\_\_\_\_。

#### 四、问答题

1. 简述a标签4个伪类书写顺序的原因。

2. 简述的alt与title的区别。

3. 简述href属性与src属性的区别。

4. 简述overflow: auto与overflow: scroll的区别。

5. 谈谈你对margin负值的理解。

#### 五、基础类代码题

设置一个a标签,文本内容为“超链接”,当光标移入时显示“HTML5 学堂”的提示信息,当单击a标签时,在一个新页面打开<http://www.h5course.com>的链接地址。

### 17.6.2 习题答案

#### 一、单项选择题

1. 【应用】A

2. 【应用】D

#### 二、判断题

1. 【理解】错。(如果页面中有id名为##的元素,会发生变化)

2. 【记忆】对。

3. 【理解】对。

4. 【理解】错。(根据具体需求而定)

5. 【理解】错。(实际上margin负值的方式要比:first-child这种特殊选择器的操作方式更加复杂,难度也比较大,只是margin负值做到了IE6的兼容。对于IE7+的各类浏览器,:first-child选择器的方法会更加简单)

#### 三、填空题

1. 【考】【记忆】:link、:visited、:hover、:active

2. 【记忆】cursor: pointer;

3. 【考】【记忆】3~5 像素空隙

#### 四、问答题

1. 【考】【理解】略,请查阅第8章的具体知识

2. 【考】【记忆】略,请查阅第8章的具体知识

3. 【考】【理解】略,请查阅第8章的具体知识

4. 【考】【记忆】略,请查阅第8章的具体知识

5. 【考】【理解】

答题提示:可以从为何使用margin负值(考虑后台数据条数)开始谈起,之后讲解基本原理(多嵌套一层元素,之后,最内层的每个元素样式统一,通过为中间层元素设置margin负值,让某个“分隔线”(也可以是别的)移出了外层元素的内容区域,之后,为最外层元素设

置 overflow 属性,实现隐藏)。

## 五、基础类代码题

### 【考】【应用】

```
<a href="http://www.h5course.com" target="_blank" title="HTML5 学堂">超链接</a>
```



## 17.7 习题集 07

### 17.7.1 习题内容

对应章节:第7章

#### 文本等细节类样式处理

涉及的主要知识:

- 背景类样式;
- 透明度的处理方法;
- 背景图合并;
- 段落类样式;
- 字体类样式。

#### 一、单项选择题

1. 当为一个 div 元素设置背景图 background: url(XXX) center no-repeat;时,表示( )。  
A. background-position 水平方向居中,垂直方向在顶部  
B. background-position 垂直方向居中,水平方向在左边  
C. background-position 水平和垂直方向均为居中  
D. 这种书写方法会导致 CSS 代码无法解析
2. 采用背景图合并的原因是( )。  
A. 使用 background-position 便于控制图片位置  
B. 更加方便后期维护  
C. 减少服务器请求次数,提升加载速度  
D. 合并后的图片大小要小于多张小图片的总和
3. 拥有 italic 的默认样式的标签是( )。  
A. strong                      B. span                      C. b                      D. em
4. text-decoration 属性的属性值中不包括( )。  
A. underline                  B. centerline                  C. overline                  D. none
5. 【多选】使用谷歌浏览器的 F12 查看元素的盒模型特质时,盒模型中的某个属性出现了小数,有可能是( )。  
A. 子级设置了 margin: 0 auto;,子级和父级之间的空隙为奇数  
B. 书写代码的时候写了小数,如 width: 210.5px;  
C. 浏览器被放大或缩小  
D. 不会出现小数,所有的小数都会四舍五入转换成整数
6. 将“< span > HTML5 学堂</span >”当中“HTML5”这几个字母的间距调整大一点儿的方法是( )。  
A. 调整 letter-spacing 的值                      B. 调整 letter-padding 的值



- C. 调整 word-spacing 的值                      D. 调整 word-padding 的值
7. 将“< div> HTML……(重复 100 次 HTML) </div>”,如下说法不正确的是( )。
- A. 不能够换行,一直在一行当中显示,且超出容器
- B. 能够正常地显示和换行
- C. 可以通过 word-break: break-all;强制内容换行
- D. 可以通过 word-wrap: break-word;让内容强制换行

## 二、判断题

1. 为了保证用户的体验效果,将网页中的最小字体设置为 12 像素,小于 12 像素的字体在浏览器中是看不到的。
2. 将字体设置为 15px 时,文字在谷歌浏览器中显示时,字体大小会变化为 14px。
3. font 是一种复合属性,能够将多种字体类样式书写在一起,也能够根据需求灵活运用,如需要设置加粗和字体大小以及行高,就可以通过 font 直接设置这三者来完成。
4. 使用 text-indent,主要用于段落的缩进控制,可以为其设置负值。
5. 使用 text-indent,除了能够使用 px 单位之外,还能够使用 em,但是不允许使用百分比(%)。使用的单位当中,px 没有 em 灵活。

## 三、填空题

1. 浏览器常用字体包括 \_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_ (三种中文字体,三种英文字体,请书写单词)。
2. 控制超链接(a 标签)打开目标的属性是: \_\_\_\_\_,其属性值包括 \_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
3. 网页的默认字体大小是 \_\_\_\_\_。
4. 在默认浏览器下,IE、火狐、谷歌浏览器的行高默认为 \_\_\_\_\_ ~ \_\_\_\_\_ px。

## 四、问答题

1. 简述实现背景透明的方法以及区别。
2. 简述背景图合并的实现原理,以及与 CSS Sprite 的区别。
3. 简述 word-wrap 与 word-break 的区别。
4. 简述 word-spacing 与 letter-spacing 的区别。
5. 简述 italic 与 oblique 的区别。

## 五、基础类代码题

请根据需求实现如下效果。

HTML5 学堂,致力打造前端、HTML5 的原创分享平台。由利利和堡堡创建,当前由多名开发工程师和几名讲师组成,在这里,95% 的文章都出自学堂团队,涵盖了 HTML、CSS、JavaScript、jQuery、AJAX 等各类前端知识,还有 HTML5 的实例开发,对于 JS 类库、JS 底层知识、前端的相关技术以及行业的未来发展等都有所涉及。在这里,技术可以通俗易懂,我们希望大家一起将 HTML5 学堂打造成个性化学习网站,为您提供最适合您的信息。

需求:

整个段落元素,宽度 400 像素,1 像素的红色边框,在 body 当中水平居中,段落顶部距

离 body 元素有 20 像素的空隙；

段落与具体内容之间,4 个方向各有 10 像素的距离；

段落文本首行缩进、字体为微软雅黑、字体大小为 14 像素、行高值 18 像素、水平方向两端对齐；

段落文本当中的“HTML5 学堂”是一个链接,显示颜色为浅蓝(#39f),存在下画线；

段落文本当中的“原创分享平台”、“技术可以通俗易懂”的字样,设置为红色、加粗。

## 17.7.2 习题答案

### 一、单项选择题

1. 【记忆】C
2. 【理解】C
3. 【记忆】D
4. 【记忆】B
5. 【考】【记忆】AC
6. 【记忆】A
7. 【记忆】B

### 二、判断题

1. 【考】【记忆】错。(并不会消失看不到)
2. 【考】【记忆】错。(以 15px 的样式渲染)
3. 【考】【记忆】错。(font 属性必须设置字体类型、字体大小,否则 font 设置会失效)
4. 【记忆】对。
5. 【记忆】错。(可以使用百分比作为单位)

### 三、填空题

1. 【考】【记忆】Microsoft YaHei(微软雅黑)、SimSun(宋体)、SimHei(黑体)、Arial、sans-serif、Tahoma
2. 【考】【记忆】target、\_blank、\_self、\_parent、\_top
3. 【考】【记忆】12px
4. 【考】【记忆】19~22px

### 四、问答题

1. 【考】【记忆】略,请查阅第 7 章的具体知识
2. 【考】【理解】略,请查阅第 7 章的具体知识
3. 【考】【记忆】略,请查阅第 7 章的具体知识
4. 【考】【记忆】略,请查阅第 7 章的具体知识
5. 【考】【记忆】略,请查阅第 7 章的具体知识

### 五、基础类代码题

#### 【应用】

```
<!doctype html >  
<html >
```



```

< head>
  < meta charset = "UTF - 8">
  < title>HTML5 布局之路 - 文本类样式 - 基础代码题</title>
  < link rel = "stylesheet" href = "../css/reset.css">
  < style>
    .exam {
      width: 400px;
      padding: 10px;
      margin: 20px auto 0;
      border: 1px solid red;
      text-align: justify;
      text-indent: 2em;
      font-family: 'Microsoft YaHei';
      font-size: 14px;
      line-height: 18px;
    }
    .exam a {
      color: #39f;
      text-decoration: underline;
    }
    .exam span {
      color: red;
      font-weight: bold;
    }
  </style>
</head>
< body>
  < p class = "exam">< a href = "http://www.h5course.com" title = "">HTML5 学堂</a>, 致力打
  造前端、HTML5 的< span>原创分享平台</span>. 由利利和堡堡创建, 当前由多名开发工程师和几名讲
  师组成, 在这里, 95 % 的文章都出自学堂团队, 涵盖了 HTML、CSS、JavaScript、jQuery、AJAX 等各类前端
  知识, 还有 HTML5 的实例开发, 对于 JS 类库、JS 底层知识、前端的相关技术以及行业的未来发展等都
  有所涉及. 在这里,< span>技术可以通俗易懂</span>, 我们希望大家一起将< a href = "http://
  www.h5course.com" title = "">HTML5 学堂</a>打造成个性化学习网站, 为您提供最适合您的信息.
</p>
</body>
</html>

```



## 17.8 习题集 08

对应章节: 第8、9章

涉及的主要知识:

### 特殊布局情况

- 定位布局
- 界限控制与伪元素的妙用

- 定位布局;
- 层级覆盖关系;
- 最大最小宽度、高度;
- 伪元素及其应用。

17.8.1 习题内容

一、单项选择题

1. 如下定位方式当中,肯定不会让元素脱离文档流的是( )。
- A. absolute
- B. relative
- C. fixed
- D. inherit
2. 先查看如下代码,之后根据样式,选择相应代码显示的效果图。( )

结构代码:

```
<div class = "wrap">
  <div class = "box">
    <div class = "con"></div>
  </div>
</div>
```

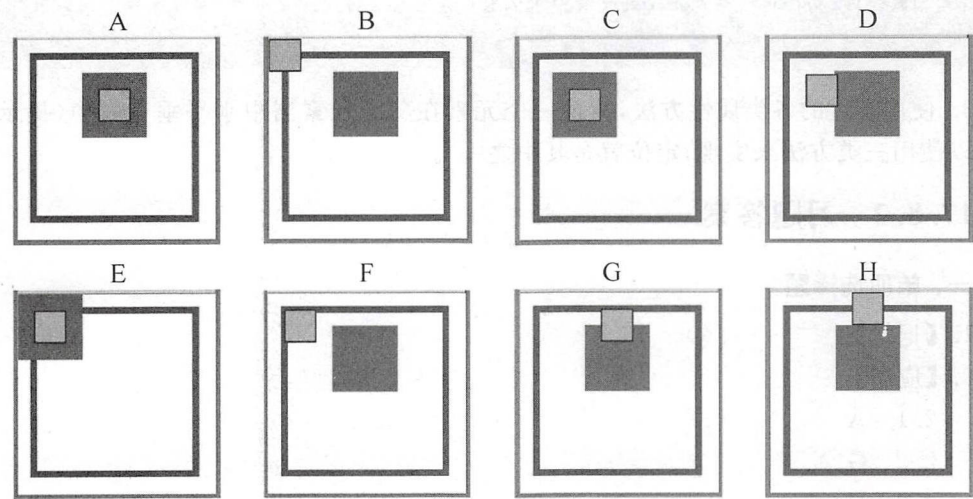
样式代码:

2.1	2.2	2.3
<pre>.wrap {   width: 200px;   height: 200px;   margin: 20px;   padding: 20px;   border: 10px solid red; } .box {   width: 100px;   height: 100px;   margin: 0 auto;   background: #39f; } .con {   position: absolute;   width: 46px;   height: 46px;   margin: 25px;   border: 2px solid black;   background: #ccc; }</pre>	<pre>.wrap {   width: 200px;   height: 200px;   margin: 20px;   padding: 20px;   border: 10px solid red; } .box {   width: 100px;   height: 100px;   margin: 0 auto;   background: #39f; } .con {   position: absolute;   top: 0;   width: 46px;   height: 46px;   margin: 25px;   border: 2px solid black;   background: #ccc; }</pre>	<pre>.wrap {   position: relative;   width: 200px;   height: 200px;   margin: 20px;   padding: 20px;   border: 10px solid red; } .box {   width: 100px;   height: 100px;   margin: 0 auto;   background: #39f; } .con {   position: absolute;   top: 0;   left: 0;   width: 46px;   height: 46px;   margin: 25px;   border: 2px solid black;   background: #ccc; }</pre>



2.4	2.5	2.6
<pre>.wrap {   position: relative;   width: 200px;   height: 200px;   margin: 20px;   padding: 20px;   border: 10px solid red; } .box {   position: relative;   width: 100px;   height: 100px;   margin: 0 auto;   background: #39f; } .con {   position: absolute;   top: 0;   left: 0;   width: 46px;   height: 46px;   margin: 25px;   border: 2px solid black;   background: #ccc; }</pre>	<pre>.wrap {   position: relative;   width: 200px;   height: 200px;   margin: 20px;   padding: 20px;   border: 10px solid red; } .box {   position: fixed;   width: 100px;   height: 100px;   margin: 0 auto;   background: #39f; } .con {   position: absolute;   top: 0;   left: 0;   width: 46px;   height: 46px;   margin: 25px;   border: 2px solid black;   background: #ccc; }</pre>	<pre>.wrap {   position: relative;   width: 200px;   height: 200px;   margin: 20px;   padding: 20px;   border: 10px solid red; } .box {   position: fixed;   top: 0;   left: 0;   width: 100px;   height: 100px;   margin: 0 auto;   background: #39f; } .con {   position: absolute;   top: 0;   left: 0;   width: 46px;   height: 46px;   margin: 25px;   border: 2px solid black;   background: #ccc; }</pre>

效果图：



## 二、判断题

1. 设置 position: relative; 属性的元素, 能够通过 top、left 等控制元素位置, 但是该元素并没有脱离文档流。
2. 伪元素是伪类选择器的另一个叫法, 语法是使用一个冒号“:”。
3. 如果不设置伪元素的 content 属性, 伪元素并不能够显示在页面当中。

## 三、填空题

1. 定位的属性值包括 \_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_ 5 种, 其中 \_\_\_\_\_ 是针对浏览器进行定位, 也称为固定定位, 该属性值在 \_\_\_\_\_ 浏览器下不兼容。
2. 常见的伪元素包括 \_\_\_\_\_、\_\_\_\_\_。

## 四、问答题

1. 当元素设置绝对定位之后, 到底针对谁进行定位?
2. 请总结最大宽度/高度、最小宽度/高度的使用情景。

## 五、基础类代码题

1. 如下代码中, div1 和 div2 哪个的层次级别高?

```
<div class="box1"> ——z-index 值设置为 2
  <div>div1</div> ——z-index 值设置为 3
</div>
<div class="box2"> ——z-index 值设置为 1
  <div>div2</div> ——z-index 值设置为 4
</div>
```

2. 如下代码中, div1 和 div2 哪个的层次级别高?

```
<div class="box1"> ——z-index 值设置为 2
  <div>div1</div> ——z-index 值设置为 3
</div>
<div class="box2"> ——z-index 值设置为 2
  <div>div2</div> ——z-index 值设置为 2
</div>
```

3. 使用学过的各类属性方法, 实现一个元素在父级元素当中水平垂直居中(提示: 至少可以使用三类方法来实现, 定位就是其中之一)。

## 17.8.2 习题答案

### 一、单项选择题

1. 【记忆】B
2. 【应用】
  - 2.1 A
  - 2.2 G
  - 2.3 D



2.4 A

2.5 C

2.6 E

## 二、判断题

1. 【记忆】对。
2. 【记忆】错。(伪元素和伪类选择器并不相同)
3. 【记忆】对。

## 三、填空题

1. 【考】【记忆】absolute、relative、fixed、inherit、static; fixed、IE6
2. 【考】【记忆】:after、:before

## 四、问答题

1. 【考】【理解】略,请查阅第8章的具体知识
2. 【考】【应用】略,请查阅第9章的具体知识

## 五、基础类代码题

1. 【应用】div1
2. 【应用】div2
3. 【考】【应用】

结构代码:

```
<div class = "wrap">
  <div class = "box"> HTML5 布局之路</div>
</div>
```

方式 1: 利用定位

方法 1: 为子级元素设置绝对定位 position

```
.wrap {
    position: relative;
    width: 400px;
    height: 400px;
    background: # fcf;
}
.box {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    width: 200px;
    height: 200px;
    margin: auto;
    background: # 999;
}
```

方法原理讲解：

子元素针对父级元素做绝对定位处理,设置 4 个方向的定位值为 0,设置 4 个方向的 margin 值为 auto,子元素与父元素之间的空隙会被子元素的 margin 值填充,左右、上下对称。

方法 2:

```
.wrap {
    position: relative;
    width: 400px;
    height: 400px;
    background: # fcf;
}
.box {
    position: absolute;
    left: 50%;
    top: 50%;
    width: 200px;
    height: 200px;
    margin: -100px 0 0 -100px;
    background: # 999;
}
```

430

方法原理讲解：

子元素进行定位(使用 50% 的相对度量单位),之后运用 margin 负值,将元素向左向上移动,移动的距离是该子元素的宽度、高度的 50%。

方式 2: 利用 padding 内边距

```
.wrap {
    width: 200px;
    height: 200px;
    padding: 100px;
    background: # ccf;
}
.box {
    width: 200px;
    height: 200px;
    background: # 000;
}
```

方式 3: 利用 margin 外边距

```
.wrap {
    width: 400px;
    height: 400px;
    background: # ccf;
}
.box {
```



```
float: left;
width: 200px;
height: 200px;
margin: 100px;
background: #000;
}
```

方法原理讲解:

子元素进行运用 margin 负值,由于子元素的 margin 值会对父级元素的布局造成影响,会让父级元素也向下运动 100 像素,因此,为元素设置左浮动。



## 17.9 习题集 09

对应章节: 第10、11章

涉及的主要知识:

- ✿ table 各类元素以及用法;
- ✿ 基本数据表的开发与实现;
- ✿ 表格涉及的 CSS 属性;
- ✿ 表单常用元素及属性;
- ✿ 属性选择器;
- ✿ 表单元素的兼容问题。

### table 表格 ; 表单

431

#### 17.9.1 习题内容

##### 一、判断题

1. col 元素可以嵌套于 colgroup 元素当中,也可以单独使用。
2. 当前开发中,对于表格的宽高边框,并不使用 width、height、border 属性进行控制,而是采用 CSS。

##### 二、填空题

1. table 的基本元素包括\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
2. table 的合并边框、跨行、跨列的属性分别是\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
3. form 元素属性中,\_\_\_\_\_属性用于设置将数据提交给“谁”,\_\_\_\_\_属性用于设置表单数据的提交方法,\_\_\_\_\_属性用于设置表单中用户输入的数据发送到服务端时采用的编码类型。

##### 三、排序题

使用 table 进行数据表的制作时,基本思路是\_\_\_\_\_。

- A. 构思数据是否有必要划分“列”
- B. 拆分数据表的结构组成
- C. 查看是否需要合并
- D. 确定是否需要使用“table”实现功能需求
- E. 确定行数以及每行的单元格个数

#### 四、问答题

1. 简述 table 布局与 HTML+CSS 布局的区别。
2. 简述 table 各个元素的嵌套规则。
3. 简述“button 标签”与“类型为 button 的 input 标签”功能上的区别。
4. 简述 form 中的 readonly 和 disabled 属性的区别。
5. 简述表单元素在实际开发当中的制作方法(如自定义文本框、调整文本域尺寸大小)。

### 17.9.2 习题答案

#### 一、判断题

1. 【记忆】对。
2. 【应用】对。

#### 二、填空题

1. 【记忆】thead、tbody、tfoot、th、tr、td、caption
2. 【记忆】border-collapse、rowspan(跨行)、colspan(跨列)
3. 【记忆】action、method、enctype

#### 三、排序题

【应用】D B A E C

#### 四、问答题

1. 【考】【理解】略,请查阅第 10 章的具体知识
2. 【考】【记忆】略,请查阅第 10 章的具体知识
3. 【考】【理解】略,请查阅第 11 章的具体知识
4. 【考】【理解】略,请查阅第 11 章的具体知识
5. 【考】【应用】略,请查阅第 11 章的具体知识



## 17.10 习题集 10

对应章节:第12章

涉及的主要知识:

停下来回头看路

● 复习PC端开发的所有HTML5相关知识。

### 17.10.1 习题内容

#### 一、问答题

1. 简述块元素与行元素的区别。
2. 简述 alt 与 title 的区别。
3. 简述 link 与 @import 的区别。
4. 简述 relative 与 absolute 的区别。



5. 简述 DIV/HTML+CSS 布局与 table 布局的区别。
6. 简述 png、gif 与 jpg 的区别。
7. 简述 CSS 引入方式的种类与区别。
8. 简述 display: none 与 visibility: hidden 的区别。
9. 简述 overflow: scroll 与 overflow: auto 的区别。
10. strong、em、i、b 标签的区别是什么?
11. 简述 table 布局和 DIV+CSS 布局的区别。

## 二、基础知识罗列

1. 如何使页面每 20 秒刷新一次?
2. 如何在新窗口当中打开 a 标签的链接?
3. CSS 选择器有哪些? 列举出来,书写合适的例子,并标注部分选择器的优先级。
4. 外边距、内边距、边框有几种书写形式? 列举说明。
5. 当布局出现问题时,有可能是什么原因造成的?
6. position 中的几种属性值有哪些? 被设置绝对定位的元素是针对什么元素进行定位?
7. table 的合并边框、跨行、跨列的属性分别是什么?
8. a 的伪类选择器的书写顺序及原因。

## 三、问答题

1. 什么是 CSS 浮动问题? 阐述你所了解的浮动以及清浮动。
2. 如何优化前端页面?
3. 表单常见的兼容问题有哪些?

## 四、基础类代码题

1. margin 的负值应用。
2. overflow: hidden 的应用。
3. 一个元素在父级元素当中实现水平垂直居中(4 种方法以上,注意先书写思路,再书写代码)。

## 17.10.2 习题答案

本章主要给出的是中型和较大型的知识点,由于本章本身即为复习章节,此处提到的所有知识在前面章节当中都能够寻找到答案,因此,在这里不再给出答案。

关于最后一题(元素在父级元素中水平垂直居中)的提示:

方法 1: 改变展示类型,然后使用 vertical-align。

方法 2: 使用父级元素的内边距。

方法 3: 使用子级元素的外边距。

方法 4: 定位。

更多方法可以查看“HTML5 学堂”(http://www.h5course.com)官网。



## 17.11 习题集 11

对应章节：第13章

涉及的主要知识：

- HTML5新增元素；
- 浏览器内核；
- CSS3选择器；
- CSS3圆角边框；
- CSS3文本阴影与盒阴影；
- CSS3背景属性。

### HTML5新标签与CSS3基础

#### 17.11.1 习题内容

##### 一、单项选择题

1. 以下关于圆角边框的描述,错误的是( )。
  - A. 能够解决在主流浏览器中弧状边框的兼容问题
  - B. 能够代替原有的一些图片,减少页面中的背景图
  - C. 元素设置圆角边框(border-radius: 50%;),元素的内容并不会受到影响,依旧会按正常情况来显示
  - D. 必须为元素设置边框,否则圆角边框的效果无法生效
2. 以下关于盒阴影的描述,错误的是( )。
  - A. 不占据物理空间
  - B. 不会被其他元素所覆盖
  - C. 能够实现边框的效果,比边框效果更强大
  - D. 能够为一个元素设置多个盒阴影
3. 以下关于 CSS3 背景的描述,错误的是( )。
  - A. 可以使用 CSS3 背景尺寸来实现各个分辨率设备背景图的自适应
  - B. 能够实现渐变的背景效果
  - C. 背景切割和背景原点的属性值和作用相同
  - D. 可以通过 rgba 设置背景颜色

##### 二、填空题

1. HTML5 新增的结构标签包括\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
2. input 新增的两种较常用属性,一种能够实现自动聚焦,属性值为\_\_\_\_\_,另一种属性为占位符,属性值为\_\_\_\_\_。
3. 能够处理背景透明的 CSS 属性包括:\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
4. CSS3 渐变包括\_\_\_\_\_、\_\_\_\_\_两种。

##### 三、问答题

1. 简述浏览器内核以及浏览器内核前缀。
2. 简述:nth-child(n)与:nth-of-type(n)两种选择器的区别。



3. 简述文本阴影与盒阴影的区别。
4. 简述背景切割与背景原点的区别。
5. 简述背景尺寸的 cover 和 contain 的区别。

#### 四、基础类代码题

元素宽度 100 像素,高度 100 像素,5 像素的 #39f 颜色实线边框。当前希望元素显示为“圆环状”,并且元素内的文本不会超出圆环区域而显示,请书写代码。

### 17.11.2 习题答案

#### 一、单项选择题

1. 【应用】D
2. 【应用】B
3. 【应用】C(作用并不相同)

#### 二、填空题

1. 【记忆】header、nav、aside、section、footer、article、hgroup、figure、figcaption...
2. 【记忆】autofocus、placeholder
3. 【记忆】opacity、filter、rgba( )、transparent
4. 【记忆】线性渐变 linear-gradient、径向渐变 radial-gradient

#### 三、问答题

1. 【考】【记忆】略,请查阅第 13 章的具体知识
2. 【考】【理解】略,请查阅第 13 章的具体知识
3. 【考】【记忆】略,请查阅第 13 章的具体知识
4. 【考】【理解】略,请查阅第 13 章的具体知识
5. 【考】【记忆】略,请查阅第 13 章的具体知识

#### 四、基础类代码题

##### 【应用】

```
<style>
    .box {
        overflow: hidden;
        width: 100px;
        height: 100px;
        border: 5px solid #39f;
        border-radius: 50%;
    }
</style>
<div class="box">HTML5 布局之路</div>
```

备注: overflow 的设置至关重要。



## 17.12 习题集 12

对应章节：第14章

### 转战移动端（上）

#### ——百分比与rem

涉及的主要知识：

- 移动端发展史；
- 移动端设备调试方法；
- 视口viewport；
- 盒模型与line-height设置百分比的状态；
- CSS3度量单位；
- JS与rem的实现移动端开发的流程与方法。

### 17.12.1 习题内容

#### 一、单项选择题

1. 以下针对 PC 端和移动端的不同之处,描述错误的是( )。
  - A. PC 端在做传统浏览器兼容时很麻烦,移动端主要要做各类设备(系统、分辨率)的兼容
  - B. 移动端可以随意使用 CSS3 选择器、HTML5 新标签、CSS3 圆角、阴影、背景等属性
  - C. 移动端的加载速度要求比 PC 端还要快
  - D. PC 端的网页拿到移动端并不能够直接使用,在用户交互方面会有不同
2. 以下不同移动端分辨率对应的基准字体大小,有可能在 320px 屏幕下出现问题的是( )。
  - A. 320 像素分辨率- 12 像素基准字体大小
  - B. 480 像素分辨率- 24 像素基准字体大小
  - C. 640 像素分辨率- 24 像素基准字体大小
  - D. 1080 像素分辨率- 40 像素基准字体大小
3. 如下属性中,哪种属性在设置百分比作为单位时,是针对父级元素的宽度进行计算的?( )
  - A. margin-top
  - B. height
  - C. border-top-width
  - D. line-height
4. 如下针对移动端开发特点的描述中,错误的是( )。
  - A. 网络字体使用更加频繁
  - B. 能够使用盒阴影实现边框等各类特效,大大方便了像素的计算
  - C. 背景图和图像都能够通过 CSS 属性控制,实现在各个分辨率下的自适应
  - D. 基本不存在什么兼容问题

#### 二、判断题

1. 使用 Wamp 进行真机调试时,必须使 Wamp 处于在线模式,并且手机与计算机处于同一个局域网。
2. 边框宽度的单位并不能够解读百分比,当使用百分比设置边框宽度(border-width)时,浏览器并不显示边框。

#### 三、填空题

1. 移动端调试的基本原则是\_\_\_\_\_设备\_\_\_\_\_调试。



2. iPhone、iPad、Android(大多数机型)、WinPhone 的默认设备 viewport 数值分别是\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。

#### 四、问答题

简述 rem 与 em 的区别。

#### 五、排序题

使 JS 配合 rem 实现移动端网页开发的流程是\_\_\_\_\_。

- A. 使用百分比和 rem 替换 px
- B. 调整视口
- C. 使用 JS 控制基准字体
- D. 确定设计图的最小字体
- E. 按照设计图的像素进行开发
- F. 查看设计图,确定需要兼容的分辨率

#### 六、基础类代码题

1. 请设置移动端视口,宽度为设备宽度大小,不允许缩放,初始缩放比例为 1。
2. 如下代码运行之后,p 元素中的字体大小是多少像素? 行高是多少像素?

结构代码:

```
<body>
  <div class="box">
    <div class="con">HTML5 布局之路</div>
  </div>
</body>
```

样式代码:

```
<style>
  html {
    font-size: 24px;
  }
  body {
    font-size: 2rem;
  }
  .box {
    font-size: 0.75rem;
  }
  .con {
    font-size: 0.67em;
    line-height: 150%;
  }
</style>
```

## 17.12.2 习题答案

### 一、单项选择题

1. 【理解】C。(都要求尽可能提升加载速度,但是 PC 端和移动端没有可比性)

2. 【应用】D。(1080 像素分辨率下,基准字体应大于等于 42 像素)
3. 【记忆】A。
4. 【理解】D。(兼容问题严重,但主要是针对不同设备、内核、分辨率)

## 二、判断题

1. 【应用】对。
2. 【应用】错。(说法不严谨,在分开设置 border-width、border-color、border-style 时,如果为 border-width 设置了百分比作为单位,浏览器不会解读 border-width,但会按照默认像素进行渲染,依旧会显示边框)

## 三、填空题

1. 【记忆】多台、真机
2. 【考】【记忆】980、1024、980、1024

## 四、问答题

【考】【记忆】略,请查阅第 14 章的具体知识

## 五、排序题

【应用】F B D E A C

## 六、基础类代码题

1. 【考】【应用】

```
<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
```

2. 【应用】

字体大小: 12 像素(约为 12px,存在小数点: 12.06)。

行高大小: 18 像素。

字体计算:  $24\text{px} \times 0.75 \times 0.67 \approx 12$ 。

rem 是针对 HTML 文档的字体大小进行计算,em 是针对父级字体大小进行计算; line-height 是针对当前元素的字体大小计算。



## 17.13 习题集 13

### 17.13.1 习题内容

对应章节: 第15章

#### 转战移动端(下)

#### —— 响应式&移动端的探索

涉及的主要知识:

- ✿ 响应式布局;
- ✿ 媒体查询;
- ✿ flexible.js实现“高清”移动端开发;
- ✿ 固定像素实现移动的开发;
- ✿ 移动端的兼容问题。



### 一、填空题

响应式布局的核心技术包括\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。

### 二、判断题

1. flexible.js 所采用的方法与前一章介绍的 JS 方法类似,前者是通过初始缩放比例和 width,设置视口为设备的实际分辨率宽度,后者是通过 width 属性,设置视口为设备的物理宽度。

2. 使用 MetaHandler.js,让移动端也可以使用固定像素进行开发,其基本原理是通过 initial-scale 属性,将书写的页面进行等比例缩放。

### 三、问答题

1. 简述响应式布局的优劣势。
2. 请整理移动端的兼容问题。

### 四、基础类代码题

为大于等于 500 像素的设备设置样式,CSS3 的媒体查询应当如何书写?

## 17.13.2 习题答案

### 一、填空题

【记忆】通过 meta 调整视口、通过媒体查询加载相应样式、使用相对单位替换绝对单位。

### 二、判断题

1. 【理解】对。
2. 【理解】对。

### 三、问答题

1. 【考】【理解】略,请查阅第 15 章的具体知识。
2. 【考】【应用】略,请查阅第 15 章的具体知识。

### 四、基础类代码题

【应用】@media all and (min-width:500px) { ... }



## 17.14 习题集 14

### 17.14.1 习题内容

对应章节:第16章

#### CSS3变形与动画

涉及的主要知识:

- ◆ CSS3二维变形 transform;
- ◆ CSS3三维变形、视角;
- ◆ CSS3过渡 transition;
- ◆ CSS3动画 animation。

### 一、多项选择题

1. 声明当前变形方式的种类可以通过( )。(多选)



- A. 为 body 设置-webkit-perspective 属性
  - B. 为变形元素添加:-webkit-transform-style: preserve-3d;
  - C. 为变形元素添加:-webkit-transform-style: preverse-3d;
  - D. 直接使用 CSS3 三维变形语法
2. 如下代码设置错误的是( )。(多选)
- A. rotate3d(30deg, 20deg, 30deg);
  - B. translate3d(200px, 100px, 200px);
  - C. skew3d(5deg, 5deg, 5deg);
  - D. scale3d(1, 0.8, 1.2);

## 二、判断题

- 1. 旋转后的元素会脱离文档流,从而对父级元素、网页的布局造成一定的影响。
- 2. z 轴始终垂直于计算机设备,指向使用者。
- 3. transform 的 4 种变形方式均可为负值。
- 4. 过渡(transition)实现的是一种样式到另一种样式的变化,这种变化需要“触发”,不能够自身实现这种变化。
- 5. 对于一个变形来说,应用了多个变形命令,这些命令的书写顺序不同,最终效果一定不同。

## 三、填空题

- 1. 当视角设置为 0 时表示“眼睛”与物体的距离是\_\_\_\_\_。
- 2. 设置元素背面不可视的代码是\_\_\_\_\_。
- 3. 二维变形中,可以通过\_\_\_\_\_命令改变元素的变形中心。
- 4. 二维变形默认旋转正方向为\_\_\_\_\_时针。
- 5. 当为 body 元素设置三维变形时,会对\_\_\_\_\_定位方式产生影响。
- 6. 可以通过设置动画循环属性的值为\_\_\_\_\_,来实现动画的无限次数循环。
- 7. 在 transition 的过渡属性当中,可以通过使用\_\_\_\_\_属性值,实现所有样式的过渡变化。
- 8. 使用 transition(合写方式)书写过渡代码时,书写顺序为:\_\_\_\_\_,\_\_\_\_\_,\_\_\_\_\_、\_\_\_\_\_。

## 四、问答题

- 1. 三维变形与二维变形的区别是什么?
- 2. 如何理解关键帧?
- 3. 简述 CSS3 动画与 CSS3 过渡的区别。

## 五、基础类代码题

请用代码实现如下需求。

初始状态下:

- (1) 元素宽高各为 250 像素;
- (2) 元素在 body 当中水平居中,纵向外边距为 60 像素;
- (3) 元素下内边距为 15 像素,其余三个方向均为 10 像素;





- (4) 元素背景为白色,1 像素 # bfbfbf 颜色的实线边框;
- (5) 存在阴影,向右、向下各偏移 2 像素,并有 3 像素的模糊值,阴影颜色为 rgb(135, 139, 144),颜色透明度为 0.4;
- (6) 光标移入时存在小手状态;
- (7) 初始状态在如上属性设置情况下,需进行二维变形处理,缩放为原来的 80%,顺时针旋转 10°。

光标移入后的样式:

- (1) 边框颜色为 # 9a9a9a;
- (2) 5 像素的圆角边框;
- (3) 阴影,向右、向下各偏移 15 像素,20 像素的模糊值,阴影颜色为 rgb(50, 50, 50),颜色透明度为 0.4;
- (4) 二维变形,恢复角度为 0°,恢复缩放,还原为初始状态。

光标移入时,对过渡的要求:

当光标移入时发生过渡,两种状态下共同涉及的所有属性均发生变化,过渡时间为 1s,过渡方式为先慢后快,无延时。

光标移入前后的效果如图 17.2 所示。



图 17.2 光标移入前后的效果

## 17.14.2 习题答案

### 一、单项选择题

1. 【记忆】BD。
2. 【记忆】AC。(rotate3d(1.5, 1, 1.5, 20deg); skew 没有 skew3d 的命令)

### 二、判断题

1. 【记忆】错。(不会脱离文档流,但是有可能内容会与兄弟其他元素重合,或超出父级显示区域)
2. 【理解】错。(垂直于元素 x 与 y 轴构成的平面向上,当元素发生三维旋转等变化时,会跟随变化)
3. 【记忆】对。
4. 【理解】对。



5. 【理解】错。(有可能不同,但是有可能一样,最重要的是变形方式是否影响了坐标系,通常旋转和缩放都会影响坐标系)

### 三、填空题

1. 【考】【记忆】无限远
2. 【记忆】-webkit-backface-visibility: hidden;
3. 【记忆】transform-origin
4. 【记忆】顺
5. 【考】【记忆】fixed(固定)
6. 【记忆】infinite
7. 【记忆】all
8. 【记忆】过渡属性、过渡时间、过渡方式、过渡延迟时间

### 四、问答题

1. 【考】【记忆】略,请查阅第 16 章的具体知识
2. 【理解】略,请查阅第 16 章的具体知识
3. 【考】【理解】略,请查阅第 16 章的具体知识

### 五、基础类代码题

#### 【考】【应用】

代码实例:

```
<!doctype html >
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 过渡</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .pic {
      width: 250px;
      height: 250px;
      margin: 60px auto;
      padding: 10px 10px 15px;
      background: white;
      border: 1px solid #bfbfbf;
      cursor: pointer;
      box - shadow: 2px 2px 3px rgba(135, 139, 144, 0.4);
      transform: scale(0.8) rotate(10deg);
      transition: all 1s ease - in;
    }
    .pic img {
      display: block;
      width: 100%;
      height: 100%;
    }
  </style>
</head>
<body>
  <div>
    <img alt="A small image placeholder for the .pic class." data-bbox="235 830 315 895"/>
  </div>
</body>
</html>
```





```
.pic:hover {  
    border-color: #9a9a9a;  
    border-radius: 5px;  
    box-shadow: 15px 15px 20px rgba(50, 50, 50, 0.4);  
    transform: scale(1) rotate(0deg);  
}  
</style>  
</head>  
<body>  
    <div class="pic">  
          
    </div>  
</body>  
</html>
```



# 第18章

## 各章节代码实战

出于加强知识应用能力的考虑,针对前 16 章的大部分章节(第 1、11、12、14 章之外),均设置了与之相对应的代码实战题。本章所有的代码实战都比较小型,适用于提升某章知识的应用能力。对于网页实例的练习与开发,请查看第 19 章。

进行本章练习之前,建议先完成相应章节理论知识的学习,以及第 17 章中,与相应章节对应的基础知识习题。



### 18.1 代码检错——[第 2、3 章]

本节共包括两段代码检错,分别针对 HTML 与 CSS 代码。建议完成第 2 章的学习之后,进行“HTML 代码检错”练习,在完成第 3 章的学习之后进行“CSS 代码检错”练习。

#### 18.1.1 实战题目

##### 1. HTML 代码检错

###### 1) 涉及知识

HTML 文件的基本组成,HTML 头部的各类标签,字符编码。

###### 2) 功能需求

请找出如图 18.1 所示代码段当中存在的代码编写问题。

```
1 <!doctype>
2 <html>
3 <head>
4     <title>HTML5布局之路</title>
5     <meta charset="UTF-8"></meta>
6     <link rel="stylesheet" href="../css/reset.css">
7     <div class="wrap"></div>
8 </head>
9 <body>
10     网页的具体内容
11 </body>
12 </html>
```

图 18.1 HTML 代码检错





## 2. CSS 代码检错

### 1) 涉及知识

CSS 不同的引入方式, CSS 代码书写的基本规范。

### 2) 功能需求

如图 18.2 所示这段代码出现于 HTML 标签的< head></head>当中, 请找出代码段当中存在的代码编写问题。

```
1 <link rel="stylesheet" src="../../css/reset.css">
2 <style>
3     @charset 'utf8';
4     .wrap {
5         width: 500px;
6         height: 400px,
7         margin: 20px;
8         padding: 10px;
9         border: 5px 20px;
10    }
11 </style>
```

图 18.2 CSS 代码检错

## 18.1.2 实战答案

### 1. HTML 代码检错答案

参考答案:

第 1 行, 文档声明有误, 没有声明使用哪种文档声明, 应当修改为<!doctype html>;

第 5 行, 标签使用错误, meta 属于单标签, 不能够书写为< meta></meta>, 而应该修改为< meta>或< meta />;

第 4 行与第 5 行, 标签顺序错误, 应该颠倒这两行代码, 否则标题部分有可能出现乱码现象;

第 7 行, 代码位置有误, 对于 div 等内容类标签应该放置于< body></body>标签当中。

### 2. CSS 代码检错答案

参考答案:

第 1 行, css 文件引入使用的属性有误, 不应该是 src 属性, 而应该是 href 属性;

第 3 行, 书写于< style></style>中的样式, 并不需要声明字符编码, 且字符编码为“utf-8”, 而不是“utf8”;

第 6 行, height: 400px 一句代码之后, 应当使用分号, 来表示一句代码的结束;

第 8 行, padding: 10px 一句代码之后, 分号使用了中文分号, 会导致后面的语句解析错误, 应当修改为英文分号;

第 9 行, border 的缩写格式有误, border 应当是“边框宽度”、“边框类型”、“边框颜色”的缩写。





## 18.2 为元素设置盒模型样式——[第3章]

本节共包括一道习题,建议完成第3章的学习之后,进行“盒模型样式”练习。

### 18.2.1 实战题目

#### 1. 涉及知识

HTML 标签的自身(盒模型)属性,主要包括 width、height、margin、padding、border。

#### 2. 功能需求

在 body 元素当中包括两个元素,这两个元素的样式相同,宽度×高度=250px×80px,在 body 当中水平居中,具有 23px 的纵向内边距,35px 的横向内边距,18px 的纵向外边距,1px 的黑色实线边框,如图 18.3 所示。

### 18.2.2 实战答案

参考答案:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 盒模型样式练习</title>
  <link rel="stylesheet" href="../css/reset.css">
  <style>
    body {
      padding-top: 18px;
      border: 1px solid red;
    }
    .wrap {
      width: 250px;
      height: 80px;
      margin: 0 auto 18px;
      padding: 23px 35px;
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <div class="wrap">块 1</div>
  <div class="wrap">块 2</div>
</body>
</html>
```

答案提示:

关于 div 块与 body 之间的间距,可以设置为 body 的 padding-top,也可以采用 div 块的

body

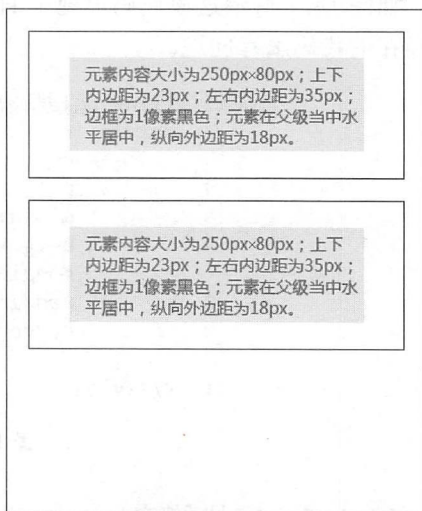


图 18.3 元素盒模型样式的设置练习





顶部外边距(margin-top)。在 div 块与块之间,纵向外边距会发生叠加,建议设置某一个方向的外边距。



## 18.3 使用浮动实现网页布局——[第 4 章]

本节共包括三道习题,建议完成第 4 章的学习之后,进行“浮动布局”练习。

### 18.3.1 实战题目

#### 1. 浮动布局练习 1

##### 1) 涉及知识

HTML 标签的盒模型属性、浮动与清浮动属性。

##### 2) 功能需求

整体内容区宽度为 1200px,在页面当中居中显示,顶部存在 10px 的黑色实线边框。每个块之间的空隙为 20px(水平、垂直),其他需求请详见图片中标注的信息,如图 18.4 所示。

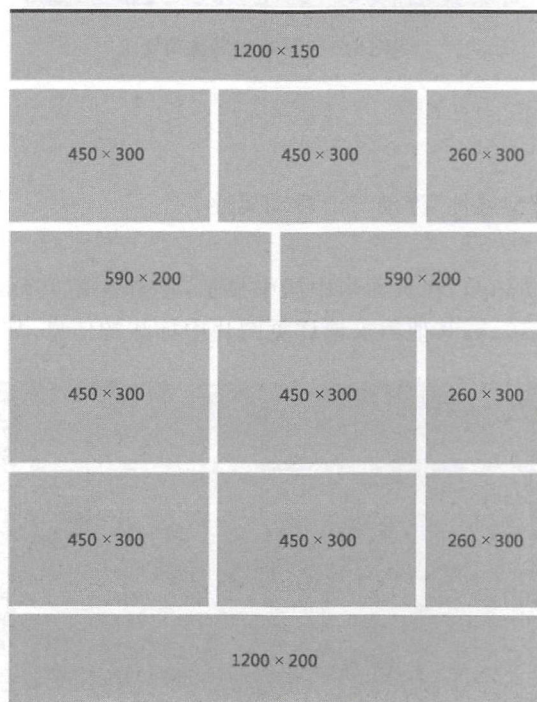


图 18.4 浮动布局练习 1

#### 2. 浮动布局练习 2

##### 1) 涉及知识

HTML 标签的盒模型属性、浮动与清浮动属性。

##### 2) 功能需求

整体内容区宽度为 960px,在页面当中居中显示,每个块之间的空隙均为 9px(水平、垂



直),其他需求请详见图片中标注的信息,如图 18.5 所示。

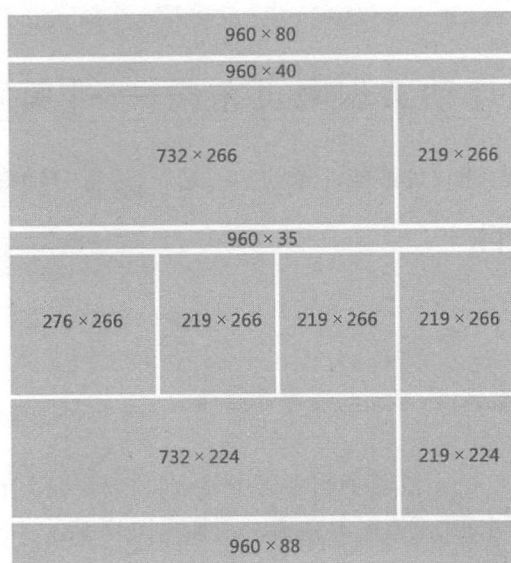


图 18.5 浮动布局练习 2

### 3. 浮动布局练习 3

#### 1) 涉及知识

HTML 标签的盒模型属性、浮动与清浮动属性。

#### 2) 功能需求

整体内容区宽度为 950px,在页面当中居中显示,左右各有 36px 的空隙。在内容部分,每个块之间的空隙均为 10px,其他需求请详见图片中标注的信息,如图 18.6 所示。

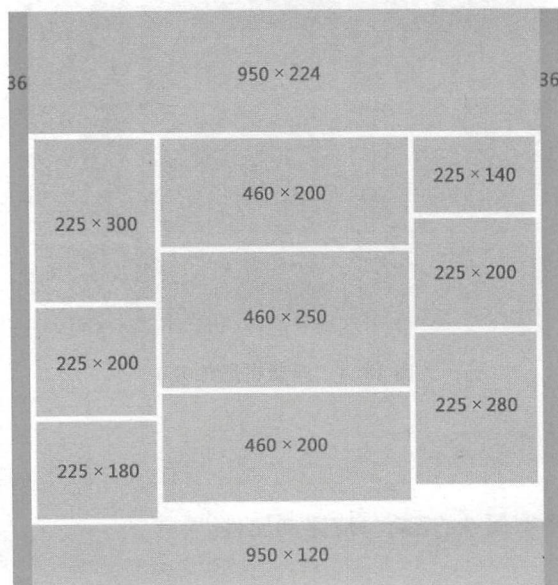


图 18.6 浮动布局练习 3





## 18.3.2 实战答案

### 1. 浮动布局练习 1 答案

参考答案：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8">
  <title>HTML5 布局之路 - 浮动布局练习 1</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .wrap {
      width: 1200px;
      margin: 0 auto;
      border-top: 10px solid #000;
    }
    .header {
      height: 150px;
      margin: 0 0 20px 0;
      background: #ccc;
    }
    .model1 {
      float: left;
      width: 450px;
      height: 300px;
      margin: 0 20px 20px 0;
      background: #ccc;
    }
    .model2 {
      float: right;
      width: 260px;
      height: 300px;
      margin: 0 0 20px 0;
      background: #ccc;
    }
    .model3 {
      float: left;
      width: 590px;
      height: 200px;
      margin: 0 0 20px 0;
      background: #ccc;
    }
    .model4 {
      float: right;
      width: 590px;
      height: 200px;
      margin: 0 0 20px 0;
```

```

        background: #ccc;
    }
    .footer {
        clear: both;
        height: 200px;
        background: #ccc;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div class="header"></div>
        <div class="model1"></div>
        <div class="model1"></div>
        <div class="model2"></div>
        <div class="model3"></div>
        <div class="model4"></div>
        <div class="model1"></div>
        <div class="model1"></div>
        <div class="model2"></div>
        <div class="model1"></div>
        <div class="model1"></div>
        <div class="model1"></div>
        <div class="model2"></div>
        <div class="model2"></div>
        <div class="footer"></div>
    </div>
</body>
</html>

```

## 2. 浮动布局练习 2 答案

参考答案：

```

<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 浮动布局练习 2</title>
    <link rel="stylesheet" href="../css/reset.css">
    <style>
        .wrap {
            width: 960px;
            margin: 0 auto;
        }
        .header {
            height: 80px;
            margin-bottom: 9px;
            background: #ccc;
        }
        .nav {

```



```
        height: 40px;
        margin-bottom: 9px;
        background: #ccc;
    }
    .pic {
        float: left;
        width: 732px;
        height: 266px;
        margin-bottom: 9px;
        background: #ccc;
    }
    .model1 {
        float: right;
        width: 219px;
        height: 266px;
        margin-bottom: 9px;
        background: #ccc;
    }
    .message {
        clear: both;
        height: 35px;
        margin-bottom: 9px;
        background: #ccc;
    }
    .model2 {
        float: left;
        width: 276px;
        height: 266px;
        background: #ccc;
    }
    .model3 {
        float: left;
        width: 219px;
        height: 266px;
        margin-left: 9px;
        margin-bottom: 9px;
        background: #ccc;
    }
    .news {
        float: left;
        width: 732px;
        height: 224px;
        margin-bottom: 9px;
        background: #ccc;
    }
    .contact {
        float: right;
        width: 219px;
        height: 224px;
```

```

        margin-bottom: 9px;
        background: #ccc;
    }
    .footer {
        clear: both;
        height: 88px;
        background: #ccc;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div class="header"></div>
        <div class="nav"></div>
        <div class="pic"></div>
        <div class="model1"></div>
        <div class="message"></div>
        <div class="model2"></div>
        <div class="model3"></div>
        <div class="model3"></div>
        <div class="model3"></div>
        <div class="news"></div>
        <div class="contact"></div>
        <div class="footer"></div>
    </div>
</body>
</html>

```

### 3. 浮动布局练习 3 答案

需求解读：

整体内容区左右的 36px 空隙，可以使用父级元素的内边距来实现；

主内容区域部分，可以设置为由内容撑开高度，这样会更有利于后期更新与维护；

内容区存在三列，可以先拆分为两部分，再进行浮动（如左侧部分包含两列，右侧部分仅包含一列）。

参考答案：

```

<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 浮动布局练习 3</title>
    <link rel="stylesheet" href="../css/reset.css">
    <style>
        .wrap {
            width: 950px;
            margin: 0 auto;
            padding: 0 36px;

```



```
        background: #39f;
    }
    .header {
        height: 224px;
        background: #999;
    }
    .main {
        padding: 10px 10px 0;
        background: #ccc;
    }
    .left-con {
        float: left;
        width: 695px;
    }
    .right-con {
        float: right;
        width: 225px;
    }
    .model1 {
        float: left;
        width: 225px;
        height: 300px;
        margin-bottom: 10px;
        background: #999;
    }
    .model2 {
        float: right;
        width: 460px;
        height: 200px;
        margin-bottom: 10px;
        background: #999;
    }
    .model3 {
        float: left;
        width: 225px;
        height: 200px;
        margin-bottom: 10px;
        background: #999;
    }
    .model4 {
        float: right;
        width: 460px;
        height: 250px;
        margin-bottom: 10px;
        background: #999;
    }
    .model5 {
        float: left;
        width: 225px;
        height: 180px;
```



```

        margin-bottom: 10px;
        background: #999;
    }
    .model6 {
        float: right;
        width: 460px;
        height: 200px;
        margin-bottom: 10px;
        background: #999;
    }
    .right-con {
        float: right;
    }
    .model7 {
        height: 140px;
        margin-bottom: 10px;
        background: #999;
    }
    .model8 {
        height: 200px;
        margin-bottom: 10px;
        background: #999;
    }
    .model9 {
        height: 280px;
        margin-bottom: 10px;
        background: #999;
    }
    .footer {
        height: 120px;
        background: #999;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div class="header"></div>
        <div class="main clearfix">
            <div class="left-con clearfix">
                <div class="model1"></div>
                <div class="model2"></div>
                <div class="model4"></div>
                <div class="model3"></div>
                <div class="model6"></div>
                <div class="model5"></div>
            </div>
            <div class="right-con">
                <div class="model7"></div>
                <div class="model8"></div>
                <div class="model9"></div>
            </div>
        </div>
    </div>
    <div class="footer"></div>

```



```
</div>
</body>
</html>
```



18.4 合理选择标签——[第 5、6 章]

本节共包括三道习题，建议完成第 5、6 章的学习之后，再进行“标签选择”的练习。

18.4.1 实战题目

1. 标签选择练习 1

1) 涉及知识

HTML 的各类标签。

2) 功能需求

解读如图 18.7 所示需求，选择合理标签，并书写布局时需要注意的事项。

◆ 经典的斐波那契数列与arguments.callee	html5学堂	2016-09-15
◆ 经典的斐波那契数列与arguments.callee	html5学堂	2016-09-15
◆ 经典的斐波那契数列与arguments.callee	html5学堂	2016-09-15
◆ 经典的斐波那契数列与arguments.callee	html5学堂	2016-09-15
◆ 经典的斐波那契数列与arguments.callee	html5学堂	2016-09-15

图 18.7 标签选择练习 1

2. 标签选择练习 2

1) 涉及知识

HTML 的各类标签。

2) 功能需求

解读如图 18.8 所示需求，选择合理标签，并书写布局时需要注意的事项。



图 18.8 标签选择练习 2

3. 标签选择练习 3

- 1) 涉及知识  
HTML 的各类标签。
- 2) 功能需求

解读如图 18.9 所示需求,选择合理标签,并书写布局时需要注意的事项。



图 18.9 标签选择练习 3

18.4.2 实战答案

1. 标签选择练习 1 答案

需求解读:

此部分是多篇文章标题的集合(即一个文章标题的列表),该集合对文章顺序并没有特定要求,每一行除了标题之外,还包括作者和时间信息,当用户单击相应行的内容时,跳转到相应的具体内容页。

参考答案:

```
<ul>
  <li>
    <a href = "" title = "">
      <strong>经典的斐波那契数列与 arguments.callee</strong>
      <span>2016 - 09 - 15 </span>
      <span>HTML5 学堂</span>
    </a>
  </li>
  <!-- <li></li>共五组 -->
</ul>
```



选用标签时应思考的事项：

如果作者信息和时间两个部分的样式相同，可以采用同样的标签（span），如果两者样式不同，则需要使用不同的标签；

出于单击区域的考虑，a 标签应当包含具体的其他标签，并且需要设置宽高（转换成块元素或设置浮动之后），由于 a 标签属于行元素，理论上最好不要包含块元素，因此，a 标签内部不适宜使用 h2 等标签，而应该采取 span、strong 等行元素。

针对 strong 标签设置左浮动之后，需要设置固定高度和宽度，且需要设置超出隐藏。

针对 span 标签设置右浮动，因此，先浮动的 span 标签会位于最右侧，因此，需要注意两个 span 的书写顺序。

## 2. 标签选择练习 2 答案

需求解读：

这是一个首页或二级页面的模块，其中包括模块的名称、模块中的几篇文章，每篇文章以“图文并茂”的方式进行呈现，包括文章的预览图、文章标题和描述信息。

参考答案：

```
<h2>
  <span>公益活动</span>
  <a href = "" title = "">more</a>
</h2>
<div>
  <dl>
    <dt><a href = "" title = ""><img src = "" alt = "" title = ""></a></dt>
    <dd>
      <h3><a href = "" title = "">网易娱乐</a></h3>
      <p>超级面对面易娱乐易级面</p>
    </dd>
  </dl>
</div>
<!-- <dl></dl>共 4 组 -->
```

选用标签时应思考的事项：

图文并茂的文章列表，适合采用 dl 标签实现布局；

当单击最顶部的 more 时，应当能够跳转到列表页，因此需要设置 a 标签；

当单击每篇文章的图像或文章标题时，应当能够跳转到内容页，因此需要设置 a 标签；

针对 dt 中的 a 标签以及 img 标签，需要处理宽高，还要防止 img 底部三像素问题；

h3 当中的 a 标签也需要处理宽高；

对于 h3 中的标题、p 标签中的内容，都需要考虑超出隐藏和超出显示为省略号。

## 3. 标签选择练习 3 答案

需求解读：

这是一个内容页面，进行的具体某一个店铺的展示，属于网站中最“深”层次的页面。

参考答案：

```
<h1>店铺名称</h1>
<dl>
  <dt><img src="" alt="" title=""></dt>
  <dd><p>此处为介绍文本</p></dd>
</dl>
<div class="inf">
  <div class="inf-left">
    <p><span>店号:</span><strong>I2-1234</strong></p>
    <p><span>电话:</span><strong>1234567890</strong></p>
    <p><span>网址:</span><strong>www.xxxxx.com</strong></p>
  </div>
  <div class="inf-right">
    <p><span>营业时间</span></p>
    <p><span>周一至周五:</span><strong>09:00 至 22:00</strong></p>
    <p><span>周六至周日:</span><strong>09:00 至 24:00</strong></p>
  </div>
</div>
<div class="pics">
  <img src="" alt="" title="">
  <img src="" alt="" title="">
</div>
<div class="more">
  <span class="pos">店铺位置</span>
  <span class="morepic">美图欣赏</span>
  <span class="about">关于我们</span>
</div>
```

选用标签时应思考的事项：

由于属于内容页，因此在标题、图像等位置，均不需要设置超链接。

页面底部的“店铺位置”、“美图欣赏”、“关于我们”，要根据网页的需求而选择标签，如果网页不要求进行跳转，而是以窗口的方式在当前页面打开内容，则此处不应该选择 a 标签，采用 span 等标签配合 JavaScript 来实现；如果要求单击之后跳转页面，则需要使用 a 标签。



## 18.5 文本样式处理——[第7章]

本节共包括一道习题，建议完成第7章的学习之后，再进行“文本样式”的练习。

### 18.5.1 实战题目

#### 1. 涉及知识

HTML 的各类标签、盒模型属性、文本类属性(字体、段落等)。

#### 2. 功能需求

书写一封书信，书信左上角的为“收信方”，右下角为“寄信方”，中间部分为正文；

浏览器中包含所有书信内容的元素，存在 1px 的实线边框，边框颜色为 #999；元素边



框与书信内容之间,在4个方向上均存在着20px的距离;

收信方字体大小为20px,加粗显示,与下面的内容区存在8px的距离;在收信方中的“To”字体大小为28px,颜色为#39f蓝色,字符间距为-2px,与右侧的“HTML5的爱好者们”拥有8px的间距;

寄信方的内容右对齐,字体大小18px;

正文部分,每个段落的行高为20px,首行缩进两个字符,存在底部外边距为8px。

效果如图18.10所示。

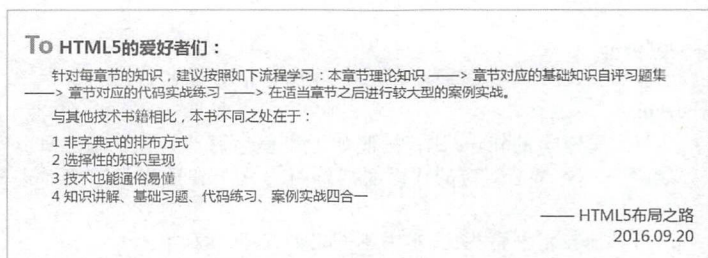


图 18.10 文本样式练习题

## 18.5.2 实战答案

参考答案:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF-8" />
  <title>HTML5 布局之路 - 文本样式练习</title>
  <link rel = "stylesheet" type = "text/css" href = "../css/reset.css" />
  <style type = "text/css">
    .wrap {
      padding: 20px;
      border: 1px solid #999;
    }
    .mailto {
      margin-bottom: 8px;
      font-size: 20px;
      font-weight: bold;
    }
    .mailto span {
      margin-right: 8px;
      font-size: 28px;
      letter-spacing: -2px;
      color: #39f;
    }
    .mailcon p {
      margin-bottom: 8px;
      line-height: 20px;
      text-indent: 2em;
    }
  </style>
</head>
<body>
  <div class = "wrap">
    <div class = "mailto">
      <span>To HTML5的爱好者们:</span>
      <p>针对每章节的知识,建议按照如下流程学习:本章节理论知识——> 章节对应的基础知识自评习题集
      ——> 章节对应的代码实战练习——> 在适当章节之后进行较大型的案例实战。</p>
      <p>与其他技术书籍相比,本书不同之处在于:</p>
      <ol>
        <li>1 非字典式的排布方式</li>
        <li>2 选择性的知识呈现</li>
        <li>3 技术也能通俗易懂</li>
        <li>4 知识讲解、基础习题、代码练习、案例实战四合一</li>
      </ol>
      <p style = "text-align: right;">—— HTML5布局之路
      2016.09.20</p>
    </div>
  </div>
</body>
</html>
```

```

        .mailcon ol {
            text-indent: 2em;
        }
        .mailfrom {
            text-align: right;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <div class="wrap">
        <p class="mailto"><span>To</span> HTML5 的爱好者们:</p>
        <div class="mailcon">
            <p>针对每章节的知识,建议按照如下流程学习: 本章节理论知识 ——> 章节对应的
            基础知识自评习题集 ——> 章节对应的代码实战练习 ——> 在适当章节之后进行较大型的案例实
            战.</p>
            <p>与其他技术书籍相比,本书不同之处在于:</p>
            <ol>
                <li>1 非字典式的排布方式</li>
                <li>2 选择性的知识呈现</li>
                <li>3 技术也能通俗易懂</li>
                <li>4 知识讲解、基础习题、代码练习、案例实战四合一</li>
            </ol>
        </div>
        <p class="mailfrom">—— HTML5 布局之路</p>
        <p class="mailfrom">2016.09.20</p>
    </div>
</body>
</html>

```



## 18.6 定位布局——[第 8 章]

本节共包括一道习题,建议完成第 8 章的学习之后,再进行“定位布局”的练习。

### 18.6.1 实战题目

#### 1. 涉及知识

HTML 的各类标签、盒模型属性、文本类属性、定位类属性。

#### 2. 功能需求

该模块是一篇攻略文章的缩略介绍,可以通过单击这个部分,进入攻略文章的具体内容页面。缩略介绍当中,包括预览图、标题、更新时间三种基本信息,在默认情况下预览图被一层半透明黑色的蒙版层(rgba(92, 92, 92, 0.3))覆盖,当光标移入时,半透明黑色的蒙版层变浅(rgba(92, 92, 92, 0.1)),如图 18.11 所示。



图 18.11 定位布局练习题



## 18.6.2 实战答案

需求解读:

整个区域都需要单击,因此需要用 a 标签包含其他各类标签;

a 标签属于行元素,原则上来说只能够包含行元素,但是,如果说考虑语义性,希望使用 a 包含块元素,也并非不可,只是从嵌套层面上不是太合理而已;

黑色蒙版层盖在图像之上,并不会影响更新时间和标题,因此,需要利用 z-index 进行等级的调整。

参考答案:

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 定位布局</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .pic {
      width: 310px;
      height: 135px;
      margin: 20px auto 0;
    }
    .pic a {
      position: relative;
      display: block;
      height: 100 % ;
      color: # fff;
    }
    .pic img {
      display: block;
      width: 100 % ;
      height: 100 % ;
    }
    .pic var {
      position: absolute;
      top: 7px;
      left: 8px;
      z - index: 1;
      width: 62px;
      height: 22px;
      background: url("../images/updatetime.png") 0 0 no - repeat;
      text - align: center;
      font - size: 12px;
      line - height: 22px;
    }
    .pic span {
      overflow: hidden;
```

```

        position: absolute;
        left: 0;
        bottom: 0;
        z-index: 1;
        width: 282px;
        height: 40px;
        margin: 0 14px;
        line-height: 40px;
    }
    .pic .mask {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background: rgba(92, 92, 92, 0.3);
    }
    .pic a:hover .mask {
        background: rgba(92, 92, 92, 0.1);
    }
}
</style>
</head>
<body>
    <div class="pic">
        <a href="http://www.h5course.com" title="">
            <var>本周更新</var>
            
            <span>此处为该攻略的标题文字,白色</span>
            <div class="mask"></div>
        </a>
    </div>
</body>
</html>

```



## 18.7 伪元素的应用——[第9章]

本节共包括两道习题,建议完成第9章的学习之后,再进行“伪元素”的练习。

### 18.7.1 实战题目

#### 1. 多重背景的设置

##### 1) 涉及知识

伪元素、各类 CSS 样式。

##### 2) 功能需求

通过 after 伪元素和 before 伪元素,实现一个“宽度自适应的元素背景”,即当一个 div 元素的宽度设置为 280px 时,为图 18.12 中上面元素的显示效果;如果将这个 div 元素的宽



度设置为 380px,则为图 18.12 中下面元素的显示效果。

## 2. 在导航项目之间添加分隔符

### 1) 涉及知识

伪元素、各类 CSS 样式。

### 2) 功能需求

使用 after 或 before 伪元素实现导航中“>”的符号,如图 18.13 所示。

块的多重背景

块的多重背景

图 18.12 伪元素设置多重背景的练习题

最新攻略 > 移动端开发 >

图 18.13 伪元素设置导航分隔符的练习题

## 18.7.2 实战答案

### 1. 多重背景答案

参考答案

```
<!doctype html >
<html >
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 伪元素应用 - 多重背景</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .common {
      position: relative;
      height: 34px;
      margin: 20px auto 0;
      padding: 0 10px;
      line - height: 34px;
      background: # eee;
    }
    .common:before {
      content: "\200B";
      position: absolute;
      left: 0;
      top: 0;
      width: 9px;
      height: 34px;
      background: url("../images/nav_left.jpg") 0 0 no - repeat;
    }
    .common:after {
      content: "\200B";
      position: absolute;
      right: 0;
      top: 0;
    }
  </style>
</head>
</html>
```

```

        width: 9px;
        height: 34px;
        background: url("../images/nav_right.jpg") 0 0 no-repeat;
    }
    .menu {
        width: 280px;
    }
    .menu2 {
        width: 380px;
    }
</style>
</head>
<body>
    <div class="menu common">块的多重背景</div>
    <div class="menu2 common">块的多重背景</div>
</body>
</html>

```

答案提示：

为元素本身设置背景颜色,使用 PS 将背景图最左端和最右端的圆角部分切下来,分别设置为 after 伪元素与 before 伪元素的背景,并将两种伪元素定位于元素的最左端和最右端。对于此处使用到的图片,可以在本书配套的电子资料当中找到。

## 2. 导航项目之间添加分隔符答案

参考答案：


```

<!doctype html >
<html >
<head >
    <meta charset="UTF-8">
    <title>HTML5 布局之路 - 伪元素应用 - 导航分隔符</title>
    <link rel="stylesheet" href="../css/reset.css">
    <style>
        .nav {
            width: 400px;
            margin: 20px auto 0;
            padding: 0 10px;
            background: #39f;
        }
        .nav a {
            float: left;
            height: 40px;
            line-height: 40px;
        }
        .nav a:after {
            content: ">";
            float: right;
            height: 40px;
        }
    </style>

```



```
padding: 0 10px;
line-height: 40px;
}
</style>
</head>
<body>
  <div class="nav clearfix">
    <a href="" title="">最新攻略</a>
    <a href="" title="">移动端开发</a>
  </div>
</body>
</html>
```



## 18.8 课程表与表格简历制作——[第 10 章]

本节共包括两道习题，建议完成第 10 章的学习之后，再进行“表格”的练习。

### 18.8.1 实战题目

#### 1. 课程表的制作与开发

1) 涉及知识

表格元素，各类 CSS 样式。

2) 功能需求

课程表共 7 列，前两列宽度各为 40 像素，后 5 列宽度各为 80 像素；  
标题高度为 28 像素，背景颜色为 #cfc，除标题的所有行高度为 24 像素；  
文字为“微软雅黑”、默认字体大小(CSS 重置文件中均有设置)，如图 18.14 所示。

×××专业课程表						
节次与星期		星期一	星期二	星期三	星期四	星期五
上年	1	高等数学		摄影摄像	大学英语	
	2					
	3	大学体育	大学英语		高等数学	
	4					
午休						
下年	5	数字电路		大学英语	C 语言	艺术基础
	6					
	7		大学语文			
	8					

图 18.14 表格练习题——课程表

2. 简易版表格简历

1) 涉及知识

表格元素, 各类 HTML 标签、各类 CSS 样式。

2) 功能需求

表格的具体列数和行数, 可以根据效果图来自行进行“表格的设计”;

简历标题, 字体大小 20 像素, 行高 40 像素, 水平垂直居中对齐;

基本信息、专业技能, 两个标题的字体大小为 16 像素, 行高 40 像素;

具体内容的字体大小为 12 像素, 行高 24 像素;

专业技能中的具体介绍信息, 行高 18 像素, 每个段落和每个段落之间, 存在 5 像素的底部外边距, 如图 18.15 所示。

求职：HTML5开发工程师——初学者

基本信息

姓名：初学者	联系电话：13010101010
年龄：24	电子邮箱：123456789@qq.com
学历：本科	求职意向：HTML5开发工程师
专业：计算机	期望薪资：面议



专业技能

对HTML&CSS技术熟练掌握, 可以良好地控制代码的可扩展性和命名规范、合理使用标签和利用标签语义性。

熟练掌握DIV+CSS技术, 能够轻松地处理或避免兼容性问题, 书写出扩展性强、易于前后台交互、精简、兼容性良好的前端页面。

能娴熟地使用CSS3的阴影、边框实现设计效果, 同时对CSS3三维立体变形、动画和过渡技术熟练掌握。

图 18.15 表格练习题——简易版简历

18.8.2 实战答案

1. 课程表的制作与开发答案

参考答案：

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - 表格</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <style>
    .course {
      margin: 20px auto;
      border: 1px solid black;
    }
    .course caption {
```



```

        height: 28px;
        border: 1px solid black;
        background: #cfc;
        text-align: center;
        line-height: 28px;
    }
    .course tr {
        height: 24px;
    }
    .course th, .course td {
        border: 1px solid black;
        text-align: center;
    }
    .course-list {
        width: 40px;
    }
    .course-con {
        width: 80px;
    }
</style>
</head>
<body>
    <table class="course">
        <caption>XXX 专业课程表</caption>
        <colgroup>
            <col class="course-list" span="2">
            <col class="course-con" span="5">
        </colgroup>
        <thead>
            <tr>
                <th colspan="2">节次与星期</th>
                <th>星期一</th>
                <th>星期二</th>
                <th>星期三</th>
                <th>星期四</th>
                <th>星期五</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td rowspan="4">上午</td>
                <td>1</td>
                <td rowspan="2">高等数学</td>
                <td></td>
                <td rowspan="3">摄影摄像</td>
                <td rowspan="2">大学英语</td>
                <td></td>
            </tr>
            <tr>
                <td>2</td>
                <td></td>
            </tr>
        </tbody>
    </table>

```

```
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td rowspan="2">大学体育</td>
<td rowspan="2">大学英语</td>
<td></td>
<td rowspan="2">高等数学</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td colspan="7">午休</td>
</tr>
<tr>
<td rowspan="4">下午</td>
<td>5</td>
<td rowspan="2">数字电路</td>
<td></td>
<td rowspan="2">大学英语</td>
<td rowspan="3">C语言</td>
<td rowspan="2">艺术基础</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td rowspan="2">大学语文</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
</body>
</html>
```







```

        .course tr {
            line-height: 24px;
        }
        .course img {
            display: block;
            width: 100%;
            height: 99px;
        }
        .course td p {
            margin: 0 0 5px;
            line-height: 18px;
        }
    </style>
</head>
<body>
    <table class="course">
        <caption>求职: HTML5 开发工程师 —— 初学者</caption>
        <colgroup>
            <col class="item1">
            <col class="inf1">
            <col class="item2">
            <col class="inf2">
            <col class="pic">
        </colgroup>
        <tbody>
            <tr class="part-tit">
                <td colspan="5">基本信息</td>
            </tr>
            <tr>
                <td>姓名: </td>
                <td>初学者</td>
                <td>联系电话: </td>
                <td>13010101010 </td>
                <td rowspan="4"></td>
            </tr>
            <tr>
                <td>年龄: </td>
                <td>24 </td>
                <td>电子邮箱: </td>
                <td>123456789@qq.com </td>
            </tr>
            <tr>
                <td>学历: </td>
                <td>本科</td>
                <td>求职意向: </td>
                <td>HTML5 开发工程师</td>
            </tr>
            <tr>
                <td>专业: </td>
                <td>计算机</td>

```





```

        <td>期望薪资:</td>
        <td>面议</td>
    </tr>
    <tr class = "part - tit">
        <td colspan = "5">专业技能</td>
    </tr>
    <tr>
        <td colspan = "5">
            <p>对 HTML&CSS 技术熟练掌握,可以良好地控制代码的可扩展性和命名规范、合理使用标签和利用标签语义性.</p>
            <p>熟练掌握 DIV + CSS 技术,能够轻松地处理或避免兼容性问题,书写出扩展性强、易于前后台交互、精简、兼容性良好的前端页面.</p>
            <p>能娴熟地使用 CSS3 的阴影、边框实现设计效果,同时对 CSS3 三维立体变形、动画和过渡技术熟练掌握.</p>
        </td>
    </tr>
</tbody>
</table>
</body>
</html>

```



## 18.9 CSS3 阴影特效——[第 13 章]

本节共包括两道习题,建议完成第 13 章的学习之后,再进行“CSS3 阴影”的练习。

### 18.9.1 实战题目

#### 1. 多种文本阴影特效

##### 1) 涉及知识

CSS3 文本阴影、各类 CSS 样式。

##### 2) 功能需求

自上而下,分别是“浮雕”、“外发光”、“模糊”、“内阴影/凹陷”、“火焰”效果文字,如图 18.16 所示。

**备注:** 为了显示效果更明显,可以选用 Britannic Bold 等宽度较宽的字体。宋体比较细,不建议使用,如果计算机中没有 Britannic Bold 字体,可以选用微软雅黑。

#### 2. 盒阴影实现多重边框

##### 1) 涉及知识

CSS3 盒阴影、各类 CSS 样式。

##### 2) 功能需求

如图 18.17 所示,从外到内分别是“红橙黄绿青蓝紫”的边框,每种边框宽度为 2 像素。各个颜色对



图 18.16 文本阴影练习题



应的 rgb 颜色值如下。

红色——rgb(255, 0, 0);  
 橙色——rgb(255, 127, 0);  
 黄色——rgb(255, 255, 0);  
 绿色——rgb(0, 255, 0);  
 青色——rgb(0, 255, 255);  
 蓝色——rgb(0, 0, 255);  
 紫色——rgb(255, 0, 255)。



图 18.17 盒阴影练习题

## 18.9.2 实战答案

### 1. 多种文本阴影特效答案

几种文本阴影特效均采用同样结构：

```
<div class="con">
  <p>HTML5 布局之路</p>
</div>
```

几种文本阴影特效均设置的通用样式：

```
.con {
  margin: 0 auto;
}
.con p {
  width: 480px;
  height: 50px;
  margin: 0 auto;
  padding: 40px 0;
  font-size: 44px;
  font-weight: bold;
  font-family: Britannic Bold;
  text-align: center;
}
```

#### 1) 浮雕效果

需求解读：

浮雕文字，针对深色的文字，在左上方设置浅色，右下方设置深色，就会有立体的感觉。建议采用深色的背景和文字，浮雕的感觉会更明显。

参考答案：

```
.con p {
  background: #454545;
  color: #333;
  text-shadow: -1px -1px #ccc, 1px 1px #666;
}
```





## 2) 外发光效果

需求解读:

外发光文字,主要是利用亮色的文字配合深色的背景,之后为文字设置阴影,重点要调整阴影的模糊度,通过多个阴影的叠加,塑造光的不同层次。

参考答案:

```
.con p {  
    background: # 000;  
    color: # fff;  
    text-shadow: 0px 0px 5px # f0f,  
                 0px 0px 10px rgba(255,0,255,0.8),  
                 0px 0px 20px rgba(255,0,255,0.8),  
                 0px 0px 35px rgba(255,0,255,0.8),  
                 0px 0px 55px rgba(255,0,255,0.8);  
}
```

## 3) 模糊效果

需求解读:

模糊文字,由于文本自身不能够模糊,而文本的阴影能够设置模糊。文本会在阴影之上,所以,首先将文本设置为 transparent(透明),再设置文本阴影(适当的模糊值)即可。

参考答案:

```
.con p {  
    background: # eee;  
    color: transparent;  
    text-shadow: 0 0 8px # 36f;  
}
```

## 4) 凹陷效果

需求解读:

凹陷文字,针对浅色的文字,在左上方设置深色,右下方设置浅色,就会有文字凹陷的感觉(与浮雕的设置方法正好相反)。建议采用浅色的背景和文字,效果会更明显。

参考答案:

```
.con p {  
    background: # d5d2c1;  
    color: # ccc;  
    text-shadow: -1px -1px # 333, 1px 1px # fff;  
}
```

## 5) 火焰效果

需求解读:

火焰文字,文字的顶部向上有“火焰”的视觉感受,由于火焰存在黄色、橙色、红色,具有一定的层次感,因此需要设置多个文本阴影,并且每个阴影都应当有一定的模糊值。



参考答案：

```
.con p {
    background: # eee;
    color: rgba(255,0,0,0.8);
    text-shadow: 0 - 3px 6px rgba(255,0,0,0.8),
                0 - 6px 10px rgba(255,64,0,0.8),
                0 - 10px 16px rgba(255,127,0,0.8),
                0 - 16px 24px rgba(255,127,0,0.4),
                0 - 22px 30px rgba(255,127,0,0.1);
}
```

## 2. 盒阴影实现多重边框答案

参考答案：

```
box-shadow: 0px 0px 0px 2px rgb(255, 0, 0),
            0px 0px 0px 4px rgb(255, 127, 0),
            0px 0px 0px 6px rgb(255, 255, 0),
            0px 0px 0px 8px rgb(0, 255, 0),
            0px 0px 0px 10px rgb(0, 255, 255),
            0px 0px 0px 12px rgb(0, 0, 255),
            0px 0px 0px 14px rgb(255, 0, 255);
```



## 18.10 响应式布局——[第 15 章]

本节共包括一道习题，建议完成第 14、15 章的学习之后，再进行“响应式布局”的练习。

### 18.10.1 实战题目

#### 1. 涉及知识

响应式布局、各类 HTML 标签、各类 CSS 样式。

#### 2. 功能需求

在不同的屏幕宽度大小下，显示效果各不相同，如图 18.18 所示。

网页内容由两部分组成，一部分为“列表内容”，另一部分是具体内容块。

在 480 及以下的分辨率屏幕当中：

body 元素的上下左右存在 5 像素的内边距；

列表内容和具体内容块，自上而下进行显示；

列表内容占满父级宽度的 100%；

每个列表项的高度为 40 像素，背景颜色为 #f0f0f0，每个列表项与列表项之间存在 5 像素的间隔；

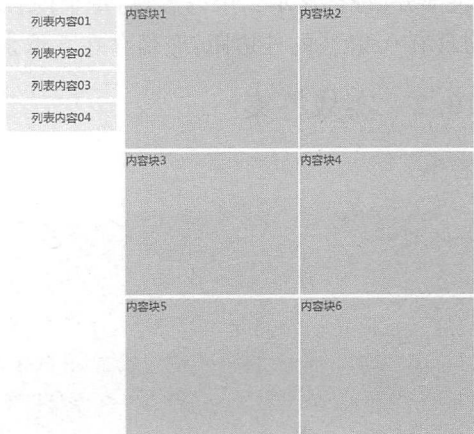
内容块部分，每行显示两个内容块，每个内容块为父级宽度的 49%，高度为 200 像素，左右分别具有 0.5% 的外边距，底部存在 5 像素的外边距，背景颜色为 #ccc。



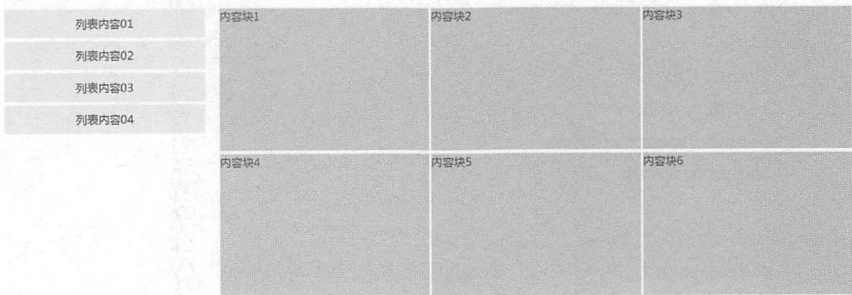




(a) 480以下分辨率的显示效果



(b) 481~960分辨率的显示效果



(c) 960以上分辨率的显示效果

图 18.18 响应式布局练习题

在 481~960 的分辨率屏幕当中：

body 元素的上下左右存在 5 像素的内边距；

左侧显示列表内容，右侧显示具体内容块；

列表内容宽度为浏览器宽度的 23.5%；

每个列表项的高度为 40 像素，背景颜色为 #f0f0f0，每个列表项与列表项之间存在 5 像素的间隔；

内容块部分，宽度为浏览器宽度的 74.5%；

内容块部分，每行显示两个内容块，每个内容块宽度为父级宽度的 49.5%，高度为 200 像素，左侧均具有 0.5% 的外边距，底部存在 5 像素的外边距，背景颜色为 #ccc。

在大于 960 的分辨率屏幕当中：

body 元素的上下左右存在 5 像素的内边距；

左侧显示列表内容，右侧显示具体内容块；

列表内容宽度为浏览器宽度的 23.5%；

每个列表项的高度为 40 像素，背景颜色为 #f0f0f0，每个列表项与列表项之间存在 5 像素的间隔；

内容块部分，宽度为浏览器宽度的 74.5%；







```

        .main div {
            width: 49.5%;
            margin-left: 0.5%;
        }
    }
    @media screen and (min-width: 961px) {
        .main div {
            width: 33%;
            margin-left: 0.33%;
        }
    }
</style>
</head>
<body>
    <div class="wrap">
        <div class="sidebar">
            <ul>
                <li>列表内容 01</li>
                <li>列表内容 02</li>
                <li>列表内容 03</li>
                <li>列表内容 04</li>
            </ul>
        </div>
        <div class="main clearfix">
            <div>内容块 1</div>
            <div>内容块 2</div>
            <div>内容块 3</div>
            <div>内容块 4</div>
            <div>内容块 5</div>
            <div>内容块 6</div>
        </div>
    </div>
</body>
</html>

```



## 18.11 二维三维特效动画——[第 16 章]

本节共包括 5 道习题,建议完成第 16 章的学习之后,再进行“二维三维特效动画”的练习。

### 18.11.1 实战题目

#### 1. 可爱的安卓机器人

##### 1) 涉及知识

CSS3 二维变形、过渡、边框。



## 2) 功能需求

使用 CSS3 将一个个 div 构建成不同的样式,之后当光标移入到安卓机器人的头部以及两个手臂的时候,让其发生旋转变化。相对比较麻烦的就是各个样式的制作以及旋转角度的计算,如图 18.19 所示。



图 18.19 可爱的安卓机器人

## 2. 晃动的文字

### 1) 涉及知识

CSS3 三维变形、过渡、动画、边框、阴影等综合效果应用。

### 2) 功能需求

相对比较复杂的效果,但均是由多个小效果组成的。当光标移入每个信息当中,当前元素背景颜色发生改变,同时,标题类字体大小变小并加粗,描述性文本字体变大,图片放大。左侧的一个淡蓝色条从左侧运动到右侧,淡蓝色条的颜色需要变深。在光标移入的时候,文字还会发生晃动效果(即从左到右再到左到右,往复),如图 18.20 所示。

## 3. 一圈一圈的呼吸灯

### 1) 涉及知识

CSS3 三维变形、动画、边框。

### 2) 功能需求

利用不同颜色的边框实现旋转,在旋转的圆环周围(可以简单地看成 loading),外侧需要有两个波纹,两个波纹之间间隔 1s,每 2s 产生一个波纹,如图 18.21 所示。

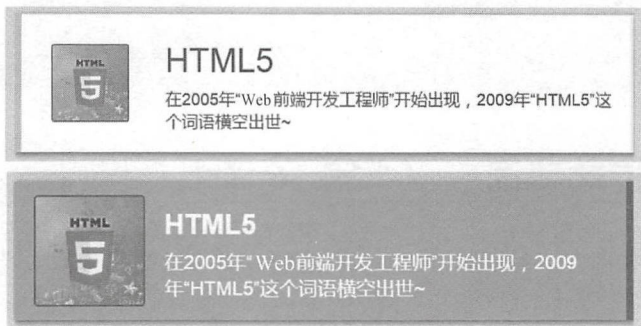


图 18.20 晃动的文字

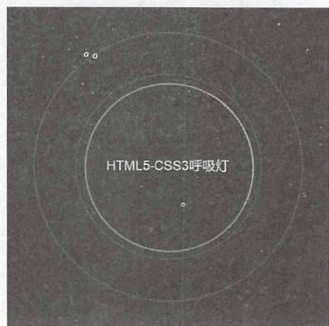


图 18.21 呼吸灯

## 4. 三维立方体

### 1) 涉及知识

CSS3 三维变形、过渡、CSS3 选择器、视角。





## 2) 功能需求

由 6 个面构建成一个立方体,每个面采用不同的半透明背景颜色,放置相应的文字信息。当光标移入的时候,立方体逐渐地发生旋转,围绕 X 轴旋转  $45^\circ$  的同时,围绕 Y 轴旋转  $45^\circ$ 。当光标移出立方体时,立方体恢复到初始状态。在最开始状态时,并不采用无限远的视角,设置一定的视角(如 1500),使得最初直视立方体时,不会觉得是一个平面,如图 18.22 所示。

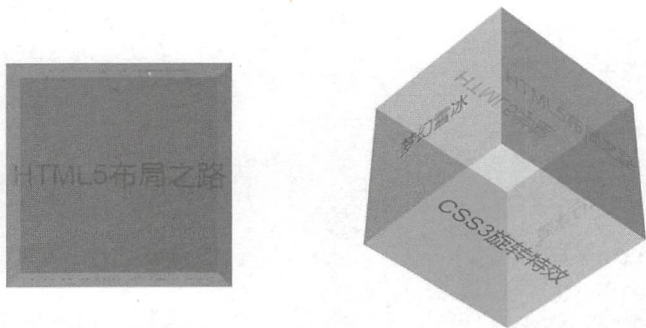


图 18.22 三维立方体

## 5. 高难度的折叠翻转卡

### 1) 涉及知识

CSS3 三维变形、过渡。

### 2) 功能需求

在光标移入时,整体会转过来,同时上下两个层也会从折叠状态变为打开状态,上面的卡先打开,下面的卡晚一会儿再打开,如图 18.23 所示。在制作的时候,要防止不同层叠到一起造成的闪屏问题。

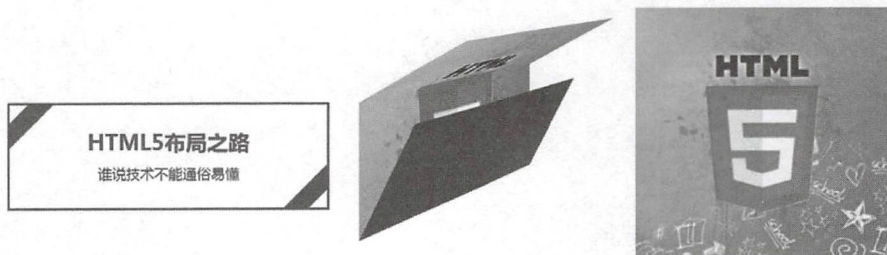


图 18.23 折叠翻转卡

## 18.11.2 实战答案

### 1. 可爱的安卓机器人答案

参考答案(出于代码量的考虑,CSS3 属性的代码设置,只给出了 WebKit 内核浏览器的书写方法):

```
<!doctype html >
<html >
```

```
< head>
< meta charset = "UTF - 8">
< title>HTML5 布局之路 - 安卓机器人</title>
< link rel = "stylesheet" href = "../css/reset.css">
< style>
    .android {
        width: 334px;
        height: 404px;
        margin: 100px auto;
    }
    .android div{
        background: # a4ca39;
        position: relative;
    }
    .android .head {
        top: 32px;
        width: 220px;
        height: 100px;
        cursor: pointer;
        border - radius: 110px 110px 0 0;
    }
    .android .l_eye, .android .r_eye {
        position: absolute;
        top: 42px;
        width: 20px;
        height: 20px;
        background: # fff;
        border - radius: 10px;
    }
    .android .l_eye {
        left: 50px;
    }
    .android .r_eye {
        right: 50px;
    }
    .android .l_ant, .android .r_ant {
        position: absolute;
        width: 6px;
        height: 50px;
        top: - 34px;
        border - radius: 3px;
    }
    .android .l_ant {
        left: 50px;
        - webkit - transform: rotate(- 30deg);
        transform: rotate(- 30deg);
    }
    .android .r_ant {
        right: 50px;
```



```
        -webkit-transform: rotate(30deg);
        transform: rotate(30deg);
    }
    .android .body{
        top: 40px;
        width: 220px;
        height: 184px;
        border-radius: 0 0 25px 25px;
    }
    .android .l_arm, .android .r_arm, .android .l_leg, .android .r_leg {
        position: absolute;
        width: 50px;
        -webkit-transition: all 0.1s ease-in;
        transition: all 0.1s ease-in;
    }
    .android .l_arm, .android .r_arm {
        height: 150px;
        border-radius: 25px;
        cursor: pointer;
    }
    .android .l_leg, .android .r_leg {
        top: 182px;
        height: 80px;
        border-radius: 0 0 25px 25px;
    }
    .android .l_arm {
        left: -58px;
    }
    .android .r_arm {
        right: -58px;
    }
    .android .l_leg {
        left: 45px;
    }
    .android .r_leg {
        right: 45px;
    }
    .android .head:hover {
        -webkit-transform: rotate(-5deg) translate(-4px, -8px);
        transform: rotate(-5deg) translate(-4px, -8px);
    }
    .android .l_arm:hover{
        -webkit-transform: rotate(15deg) translate(-14px, 0);
        transform: rotate(15deg) translate(-14px, 0);
    }
    .android .r_arm:hover{
        -webkit-transform: rotate(-30deg) translate(30px, 0);
        transform: rotate(-30deg) translate(30px, 0);
    }
}
</style>
```

```

</head>
<body>
  <div class="android">
    <div class="head">
      <div class="l_ant"></div>
      <div class="r_ant"></div>
      <div class="l_eye"></div>
      <div class="r_eye"></div>
    </div>
    <div class="body">
      <div class="l_arm"></div>
      <div class="r_arm"></div>
      <div class="l_leg"></div>
      <div class="r_leg"></div>
    </div>
  </div>
</body>
</html>

```

## 2. 晃动的文字导航答案

参考答案(出于代码量的考虑,CSS3 属性的代码设置,只给出了 WebKit 内核浏览器的书写方法):

```

<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - CSS3 晃动的导航栏</title>
  <link rel="stylesheet" href="../../css/reset.css">
  <style>
    body, html {
      height: 100%;
      min-height: 500px;
    }
    body {
      background: url('../images/cssbutton_bg.png') 0 0 repeat;
    }
    .box {
      width: 450px;
      height: 450px;
      margin: 0 auto;
      padding: 20px 0 0;
    }
    .wrap {
      position: relative;
      width: 450px;
      height: 100px;
      margin-bottom: 10px;
    }

```



```
        overflow: hidden;
        background: #FFF;
        cursor: pointer;
        box-shadow: 1px 1px 5px rgba(0, 0, 0, 0.6);
        transition: all .5s ease;
    }
    .wrap span {
        position: absolute;
        top: 0;
        left: -5px;
        width: 10px;
        height: 100px;
        background: #CCF;
        transition: all .5s ease;
    }
    .wrap img {
        float: left;
        margin: 10px 15px;
        width: 78px;
        height: 78px;
        border: 1px solid #000;
        border-radius: 3px;
        -webkit-transform: scale(0.7);
        transition: all .5s ease;
    }
    .wrap h2 {
        margin-top: 16px;
        font-size: 24px;
        transition: all 0.5s ease;
    }
    .wrap p {
        font-size: 12px;
        margin-top: 4px;
        transition: all 0.5s ease;
    }
    .wrap: hover {
        background: rgba(0, 0, 0, 0.2);
    }
    .wrap: hover span {
        left: 445px;
        background: rgba(0, 0, 255, 0.6);
    }
    .wrap: hover h2 {
        font-size: 20px;
        font-weight: bold;
        color: #FFF;
        -webkit-animation: shake 0.5s 0.2s ease;
    }
    .wrap: hover p {
        font-size: 14px;
    }
```

```

        color: #FFF;
        -webkit-animation: shake 0.5s 0.2s ease;
    }
    .wrap:hover img {
        border: 1px solid #00F;
        -webkit-transform: scale(1);
    }
    @-webkit-keyframes shake {
        0%, 100% { -webkit-transform: translateX(0); }
        20%, 60% { -webkit-transform: translateX(-10px); }
        40%, 80% { -webkit-transform: translateX(10px); }
    }
</style>
</head>
<body>
    <div class="box" id="box">
        <div class="wrap" id="con">
            <span></span>
            
            <h2>HTML5</h2>
            <p>在 2005 年"Web 前端开发工程师"开始出现,2009 年"HTML5"这个词语横空出世</p>
        </div>
    </div>
</body>
</html>

```

### 3. 一圈一圈的呼吸灯答案

参考答案(出于代码量的考虑,CSS3 属性的代码设置,只给出了 WebKit 内核浏览器的书写方法):

```

<!DOCTYPE HTML>
<html>
<head>
    <meta charset='UTF-8'>
    <title>HTML5 布局之路 - CSS3 呼吸灯</title>
    <link rel="stylesheet" href="../../css/reset.css">
    <style type='text/css'>
        .wrap {
            position: relative;
            width: 400px;
            height: 700px;
            margin: 0 auto;
            background: #000;
        }
        .circle, .change-int, .change-out, .con {
            position: absolute;
            top: 50%;

```



```

        left: 50%;
        width: 200px;
        height: 200px;
        margin: -100px 0 0 -100px;
        border: 2px solid #FF7D00;
        border-radius: 50%;
    }
    .con {
        border: none;
        color: #fff;
        font-size: 16px;
        line-height: 200px;
        text-align: center;
    }
    .circle {
        border-top-color: #E9C402;
        -webkit-animation: coor-change 1s linear infinite;
        z-index: 999;
    }
    .change-out {
        -webkit-animation: change 2s linear infinite;
    }
    .change-int {
        -webkit-animation: change 2s linear 1s infinite;
    }
    @-webkit-keyframes 'change'{
        from { -webkit-transform: scale(1); border: 1px solid #864B12;}
        to { -webkit-transform: scale(2); border: 1px solid #864B12;}
    }

    @-webkit-keyframes 'coor-change'{
        from { -webkit-transform: rotate(0deg);}
        to { -webkit-transform: rotate(360deg);}
    }
</style>
</head>
<body>
    <div class = 'wrap'>
        <div class = 'con'>HTML5 - CSS3 呼吸灯</div>
        <div class = 'circle'></div>
        <div class = 'change-out'></div>
        <div class = 'change-int'></div>
    </div>
</body>
</html>

```

#### 4. 三维立方体答案

参考答案(出于代码量的考虑,CSS3 属性的代码设置,只给出了 WebKit 内核浏览器的书写方法):

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>HTML5 布局之路 - CSS3 3D 立方体</title>
    <link rel="stylesheet" href="../css/reset.css">
    <style>
      body {
        -webkit-perspective:1500;
      }
      .main {
        position:relative;
        width:200px;
        height:200px;
        margin:100px auto;
        cursor:pointer;
        -webkit-transform-style:preserve-3d;
        -webkit-transition:-webkit-transform 2s ease 0s;
      }
      .main p {
        position:absolute;
        width:200px;
        height:200px;
        text-align:center;
        line-height:200px;
        font-size:26px;
        opacity:0.5;
      }
      .main p:nth-of-type(1) {
        background-color:red;
        -webkit-transform:translateZ(100px);
      }
      .main p:nth-of-type(2) {
        background-color:orange;
        -webkit-transform:rotateX(90deg) translateZ(100px);
      }
      .main p:nth-of-type(3) {
        background-color:yellow;
        -webkit-transform:rotateX(-90deg) translateZ(100px);
      }
      .main p:nth-of-type(4) {
        background-color:green;
        -webkit-transform:rotateY(90deg) translateZ(100px);
      }
      .main p:nth-of-type(5) {
        background-color:skyblue;
        -webkit-transform:rotateY(-90deg) translateZ(100px);
      }
      .main p:nth-of-type(6) {
        background-color:blue;
      }
    </style>
  </head>
  <body>
    <div class="main">
      <p></p>
      <p></p>
      <p></p>
      <p></p>
      <p></p>
      <p></p>
    </div>
  </body>
</html>
```



```

        - webkit - transform: rotateY(180deg) translateZ(100px);
    }
    .main: hover{
        - webkit - transform: rotateX(45deg) rotateY(45deg);
    }
</style>
</head>
<body>
    <div class = "main">
        <p>HTML5 布局之路</p>
        <p>HTML5 学堂</p>
        <p>CSS3 旋转特效</p>
        <p>独行冰海</p>
        <p>梦幻雪冰</p>
        <p>学习 HTML5 </p>
    </div>
</body>
</html>

```

答案提示:

三维立方体,是由6个面组成,每个面都是一个矩形。首先将所有的面定位在父级位置,之后进行旋转以及平移,从而构成一个立方体。在光标移入立方体时,让父级元素旋转变化。

### 5. 高难度的折叠翻转卡答案

参考答案(出于代码量的考虑,CSS3 属性的代码设置,只给出了 WebKit 内核浏览器的书写方法):

```

<!doctype html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>HTML5 布局之路 - 折叠翻转卡</title>
    <link rel = "stylesheet" href = "../css/reset.css">
    <style>
        body {
            - webkit - transform - style: preserve - 3d;
            - webkit - perspective: 1000;
        }
        .wrap {
            width: 365px;
            margin: 50px auto 0;
        }
        .main {
            float: left;
            position: relative;
            width: 363px;
            height: 363px;

```

```
        cursor: pointer;
        -webkit-transform-style: preserve-3d;
        -webkit-transform: rotateX(0deg) rotateY(180deg);
        -webkit-transition: all 1s;
    }
    .main > div {
        position: absolute;
        width: 100%;
        height: 121px;
        overflow: hidden;
        -webkit-backface-visibility: hidden;
    }
    .main img {
        display: block;
        width: 363px;
        height: 363px;
    }
    .topback, .topface {
        top: 0;
        -webkit-transform-origin: bottom;
    }
    .bottomface, .bottomback {
        top: 242px;
        -webkit-transform-origin: top;
    }
    .middleface, .middleback {
        top: 121px;
    }
    .middleface img, .middleback img {
        margin-top: -121px;
    }
    .bottomface img, .bottomback img {
        margin-top: -242px;
    }
    /* 翻转所有背面 */
    .topface {
        -webkit-transform: rotateX(-179deg);
        -webkit-transition: all 1s;
    }
    .topback {
        -webkit-transform: rotateX(-179deg) rotateY(180deg);
        -webkit-transition: all 1s;
    }
    .middleface {
        -webkit-transform: rotateX(0deg);
        -webkit-transition: all 0.7s 0.25s;
    }
    .middleback {
        -webkit-transform: rotateX(0deg) rotateY(180deg) translateZ(1px);
```



```

        -webkit-transition: all 0.7s 0.25s;
    }
    .bottomface {
        -webkit-transform: rotateX(179deg);
        -webkit-transition: all 1s 0.25s;
    }
    .bottomback {
        -webkit-transform: rotateX(179deg) rotateY(180deg);
        -webkit-transition: all 1s 0.25s;
    }
    .main:hover {
        -webkit-transform: rotateX(0deg) rotateY(0deg);
    }
    .main:hover .middleface, .main:hover .topface, .main:hover .bottomface {
        -webkit-transform: rotateX(0deg);
    }
    .main:hover .topback, .main:hover .middleback, .main:hover .bottomback {
        -webkit-transform: rotateX(0deg) rotateY(180deg);
    }
    /* 内容 */
    .con {
        position: relative;
        width: 359px;
        height: 117px;
        margin: 2px;
        background: #fff;
        text-align: center;
        overflow: hidden;
    }
    .con h1 {
        padding-top: 20px;
        font-size: 24px;
        line-height: 44px;
        font-weight: bold;
    }
    .con p {
        line-height: 28px;
    }
    .con .topsnap {
        position: absolute;
        top: 0px;
        left: -25px;
        width: 100px;
        height: 20px;
        -webkit-transform-origin: center center;
        -webkit-transform: rotate(-45deg);
    }
    .con .bottomsnap {
        position: absolute;
        bottom: 0px;



```

```

        right: -25px;
        width: 100px;
        height: 20px;
        -webkit-transform-origin: center center;
        -webkit-transform: rotate(-45deg);
    }
    /* 颜色控制 */
    .blueColor {
        background: #339;
    }
    .blueColor h1 {
        color: #009;
    }
}
</style>
</head>
<body>
    <div class="wrap">
        <div class="main">
            <div class="topface">
                
            </div>
            <div class="topback blueColor"></div>
            <div class="middleface">
                
            </div>
            <div class="middleback blueColor">
                <div class="con">
                    <h1>HTML5 布局之路</h1>
                    <p>谁说技术不能通俗易懂</p>
                    <span class="topsnap blueColor"></span>
                    <span class="bottomsnap blueColor"></span>
                </div>
            </div>
            <div class="bottomface">
                
            </div>
            <div class="bottomback blueColor"></div>
        </div>
    </div>
</body>
</html>

```

答案提示：

图像并非由一个标签组成，而是由三个标签组成。上、中、下三个块均由两个面组成（由于谷歌浏览器版本更新，在部分版本谷歌浏览器中可能无法展示成品效果）。



# 第19章

## 网页案例实战



### 19.1 PC 端网页开发实战

#### 19.1.1 功能需求

知识基础：网页开发实战包含第1~第9章的相关知识，还会涉及第13章中圆角边框的知识。本节按照网站开发的顺序，逐步进行代码的书写与讲解。

功能需求：实现图19.1的HTML与CSS代码开发。网页效果图的PSD源文件可以在本书的电子资料中下载。

兼容要求：IE9+以及各个主流浏览器正常显示，IE7、IE8浏览器中暂不考虑圆角边框类技术兼容，要保证布局不错乱。

额外备注：在该代码实例当中，所有的代码仅供参考。

#### 19.1.2 整体测绘

##### 1. 整体测绘结果

在整体测绘当中，主要进行的是大模块的测量，并不涉及具体模块内部的文字、图像等内容。整体的测绘结果如下。

(1) 设计图最大内容区宽度为990像素，在body当中水平居中，浏览器窗口大于990像素时，按照设计图给出的扩展区域背景颜色进行显示；

(2) 顶部边线，宽度占满浏览器的显示窗口，最小宽度为990像素，高度7像素；

(3) 头部(含LOGO区域和微信图标)，宽度为990像素，在父级中水平居中，高度109像素；

(4) 导航部分，宽度970像素，在父级中水平居中，高度44像素，与下面的banner(大图)部分，间隔20像素；

(5) banner(大图)部分，宽度970像素，在父级中水平居中，高度260像素，与下面的主内容区，间隔20像素；

(6) 主内容部分(含最新文章、广告)，宽度974像素，在父级中水平居中，高度774像素，左侧的最新文章部分，宽度648像素，高度774像素；右侧的广告部分，宽度262像素，高度774像素；主内容部分与下面的选项卡，间隔30像素；

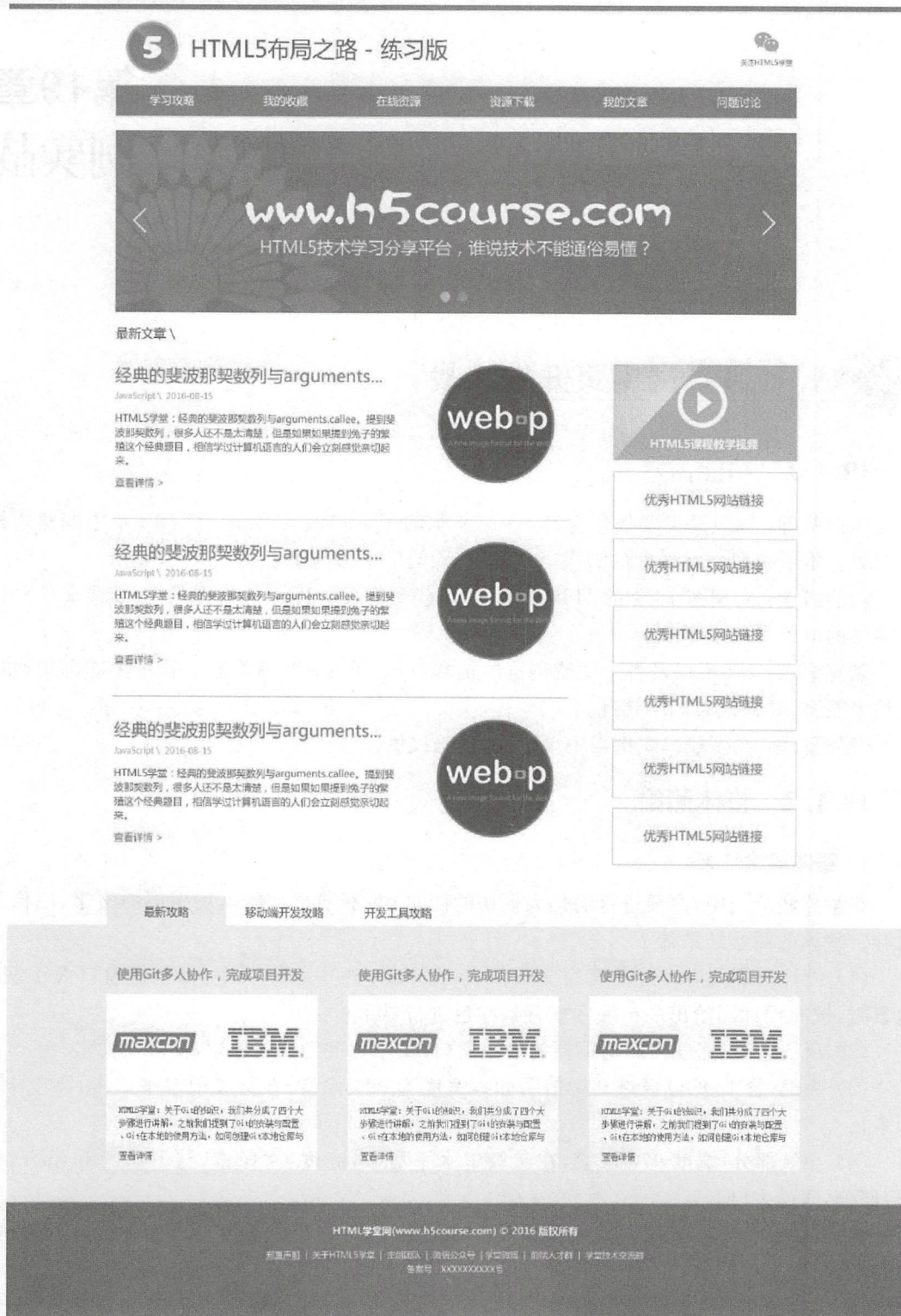


图 19.1 PC 端网页开发实战需求



- (7) 选项卡标题,宽度 990 像素,高度 42 像素;
- (8) 选项卡内容,宽度 990 像素,高度 390 像素,灰色底纹应该铺满整个浏览器;
- (9) 底部(版权)部分,宽度 990 像素,高度 165 像素,蓝色底纹铺满整个浏览器。

## 2. 测绘示意图

测绘示意图如图 19.2 所示。

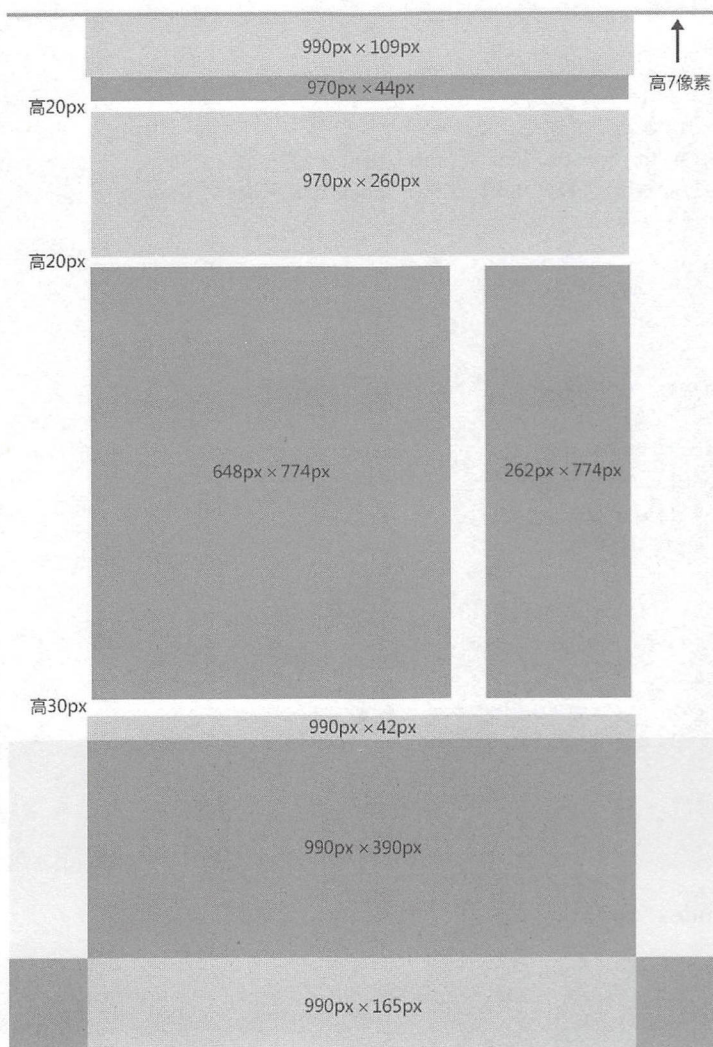


图 19.2 PC端网页开发实战整体测绘图

## 3. 调整重置文件

在该页面的设计图当中,大部分的链接颜色均为#39f蓝色,字体为微软雅黑。网页中的字体存在12px、14px、16px、18px等各种尺寸大小,因此在重置文件当中,依旧保持16px作为基准字体大小。

```
a {text-decoration:none; color:#39f; outline:none;}
```

### 19.1.3 实现整体布局

#### 1. 整体布局的相关代码

结构代码：

```
<!doctype html >
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路 - PC 端网页</title>
  <link rel = "stylesheet" href = "css/reset.css">
  <link rel = "stylesheet" href = "css/index.css">
</head>
<body>
  <div class = "header">
    <div class = "header - con"></div>
  </div>
  <div class = "nav"></div>
  <div class = "bigpic"></div>
  <div class = "con clearfix">
    <div class = "news"></div>
    <div class = "ads"></div>
  </div>
  <div class = "tabtit"></div>
  <div class = "tabcon - box">
    <div class = "tabcon"></div>
  </div>
  <div class = "footer - box">
    <div class = "footer"></div>
  </div>
</body>
</html>
```

样式代码(index.css):

```
@charset 'utf - 8';
.header {
  border - top: 7px solid #39f;
}
.header - con {
  width: 990px;
  height: 109px;
  margin: 0 auto;
}
.nav {
  width: 970px;
  height: 44px;
  margin: 0 auto 20px;
```



```
        background: #39f;
    }
    .bigpic {
        width: 970px;
        height: 260px;
        margin: 0 auto 20px;
    }
    .con {
        width: 974px;
        margin: 0 auto 30px;
    }
    .news {
        float: left;
        width: 648px;
        height: 774px;
    }
    .ads {
        float: right;
        width: 262px;
        height: 774px;
    }
    .tabtit {
        width: 990px;
        height: 42px;
        margin: 0 auto;
    }
    .tabcon-box {
        background: #f4f4f4;
    }
    .tabcon {
        width: 990px;
        height: 390px;
        margin: 0 auto;
    }
    .footer-box {
        background: #39f;
    }
    .footer {
        width: 990px;
        height: 165px;
        margin: 0 auto;
    }

    /* 如下设置只是为了显示模块,后期需要删除这些背景颜色 */
    .header-con {
        background: #ccc;
    }
    .bigpic {
        background: #ccc;
    }
}
```

```
.news {
    background: #ccc;
}
.ads {
    background: #ccc;
}
.tabcon {
    background: #ccc;
}
.tabtit {
    background: #aaa;
}
```

## 2. 代码解析与说明

在开发时,首先应当引入 reset.css 文件,以防止浏览器默认样式出现问题;

在结构的选择方面,应当精简,嵌套层数不要太深;

对于“为了标识布局块”而为布局块设置的背景颜色,建议将此类代码统一书写于所有样式代码的底部,方便后期删除。(当然,并非所有的背景颜色都需要删除,此处一定要区分清楚。)

### 19.1.4 实现头部模块(含 LOGO 与微信)部分

#### 1. 标签选择的基本分析

头部模块,包括 LOGO 和微信两个部分。

左侧的 LOGO 部分当中包含一个 LOGO 图像和“HTML5 布局之路 - 练习版”的文字。其中,图像可以作为背景图进行处理,文字属于网页的大标题,应当采用 h1 标签。

在一个网站当中,LOGO 通常是可以单击的,单击 LOGO 后,能够实现页面的跳转,因此,需要为 LOGO 设置超链接(a 标签)。

右侧的微信部分,文字大小为 10 像素,小于浏览器最小字体,因此,“关注 HTML5 学堂”的文字,应当选用“背景图”来实现。

#### 2. 标签选择结果

基于上面的基本分析,头部部分的标签选择如下。

```
<div class="header-con">
    <h1><a href="" title="">HTML5 布局之路 - 练习版</a></h1>
    <span></span>
</div>
```

#### 3. 模块基本布局样式处理

```
.header-con h1 {
    float: left;
    width: 500px;
    height: 100%;
```



```
}  
.header-con a {  
    display: block;  
    height: 100%;  
    padding-left: 93px;  
}  
.header-con span {  
    float: right;  
    width: 92px;  
    height: 100%;  
}
```

#### 4. 结合文本类样式,实现模块开发

```
.header-con h1 {  
    float: left;  
    width: 500px;  
    height: 100%;  
    line-height: 109px;  
}  
.header-con a {  
    display: block;  
    height: 100%;  
    padding-left: 93px;  
    font-size: 32px;  
    background: url('../images/bac.png') 0 0 no-repeat;  
}  
.header-con span {  
    float: right;  
    width: 92px;  
    height: 100%;  
    line-height: 109px;  
    background: url('../images/bac.png') center -109px no-repeat;  
}
```

497

#### 5. 代码解析与说明

合理利用 line-height,将行高与元素高度设置为同一个值时,能够让单行文本在元素当中垂直居中。

### 19.1.5 实现导航模块部分

#### 1. 标签选择的基本分析

导航,主要是用于链接到其他二级页面,因此需要使用 a 标签。

#### 2. 标签选择结果

基于上面的基本分析,导航部分的标签选择如下。

```
<div class="nav">
  <a href="" title="">学习攻略</a>
  <a href="" title="">我的收藏</a>
  <a href="" title="">在线资源</a>
  <a href="" title="">资源下载</a>
  <a href="" title="">我的文章</a>
  <a href="" title="">问题讨论</a>
</div>
```

### 3. 模块基本布局样式处理

```
.nav a {
  float: left;
  width: 16.66%;
  height: 100%;
}
```

### 4. 结合文本类样式,实现模块开发

```
.nav a {
  float: left;
  width: 16.66%;
  height: 100%;
  text-align: center;
  line-height: 44px;
  color: #fff;
}
```

### 5. 代码解析与说明

每个 a 标签(导航)的宽度为父级 div 宽度的 1/6,在使用百分比设置 a 标签的宽度时,不能够采用 16.67%,而应当采用 16.66%。

如果 a 标签的宽度设置为 16.67%,会导致 6 个导航的总宽度为 100.02%,超出父级宽度,此时,最后一个导航会被挤到下面。

## 19.1.6 实现内容左侧最新文章部分

### 1. 标签选择的基本分析

设计图中“最新文章”几个字,可以使用 div 标签,也可以使用 p 标签、h3 标签。

“最新文章”的下面是包含三篇文章的列表,列表属于图文并茂类,因此,应当采用 dl 标签,dl 中的 dt 包含图像,dd 包含具体文章的标题、内容等。

此处,单击图像或文字标题、查看详情,均可以实现链接跳转,因此需要在相应位置添加 a 标签,三个文章之间,存在两条灰色实线,此处可以通过伪类选择器来进行控制。

### 2. 标签选择结果

基于标签选择的基本分析,最新文章部分的标签选择如下(出于篇幅考虑,如下代码中



```
<div class = "news">
    <h3>最新文章 \</h3>
    <div class = "news - con">
        <dl>
            <dt>
                <a href = "" title = ">
                    <img src = "images/demo.png" alt = "" title = ">
                </a>
            </dt>
            <dd>
                <h2><a href = "" title = ">经典的斐波那契数列与 arguments...</a></h2>
                <div>JavaScript \&nbsp;\&nbsp;\&nbsp;2016 - 08 - 15</div>
                <p>HTML5 学堂：经典的斐波那契数列与 arguments.callee。提到斐波那契数列，
                很多人还不是太清楚，但是如果提到兔子的繁殖这个经典题目，相信学过计算机语言的人们会立刻
                感觉亲切起来.</p>
                <div><a href = "" title = ">查看详情 ></a></div>
            </dd>
        </dl>
    </div>
</div>
```

```
.news h3 {
    height: 24px;
}

.news - con dl {
    height: 219px;
    padding-top: 30px;
    border-top: 1px solid #ccc;
}

.news - con dl:first-child {
    border-top: 0;
}

.news - con dt {
    float: right;
    width: 170px;
    height: 170px;
    padding: 0 20px;
}

.news - con dt a {
    display: block;
    height: 100%;
}

.news - con dt img {
```

```

        display: block;
        width: 100%;
        height: 100%;
    }
    .news - con dd {
        float: left;
        width: 405px;
    }
    .news - con h2 {
        height: 40px;
    }
    .news - con h2 a {
        display: block;
        height: 100%;
    }
    .news - con div {
        height: 20px;
    }
    .news - con div a {
        float: left;
        height: 100%;
    }
    .news - con p {
        height: 80px;
        margin: 9px 0;
    }
}

```

#### 4. 结合文本类样式,实现模块开发

与上面代码相比,没有修改的 CSS 选择器样式并没有书写。

```

.news h3 {
    height: 24px;
    line - height: 24px;
    font - size: 18px;
    color: #555;
}
.news - con h2 a {
    display: block;
    height: 100%;
    line - height: 40px;
    font - size: 24px;
}
.news - con div {
    height: 20px;
    font - size: 12px;
    color: #b3b2b2;
}

```



```

}
.news-con div a {
    float: left;
    height: 100%;
    font-size: 14px;
}
.news-con p {
    height: 80px;
    margin: 9px 0;
    line-height: 20px;
    font-size: 14px;
    color: #666;
}

```

### 5. 代码解析与说明

将各个 dl 元素包含于 div(类名为 news-con)当中,主要目的是让 dl:first-child 选择器生效,从而去除第一个 dl 的边框;

“查看详情”部分,也可以直接使用 a 标签,而不多嵌套一层 div;

该模块是所有模块当中最为复杂的模块,针对 dd 中的各个元素进行布局时,需要利用行高配合 margin 值,来实现不同元素之间的间隔。

“行高与外边距”实现“元素间隔”的原理图,如图 19.3 和图 19.4 所示。

501

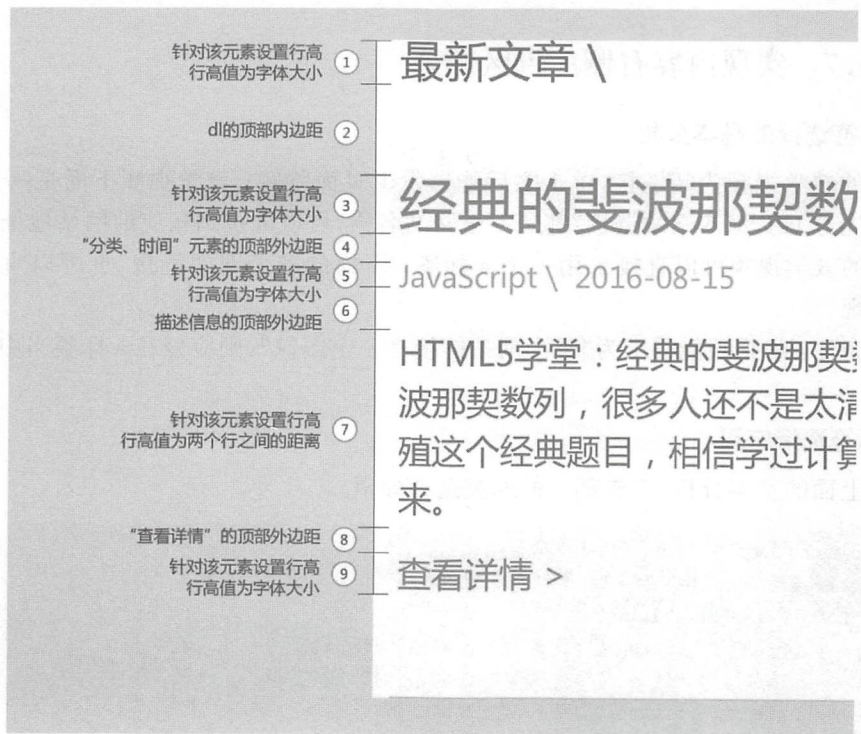


图 19.3 最新文章模块空隙设置方法 1

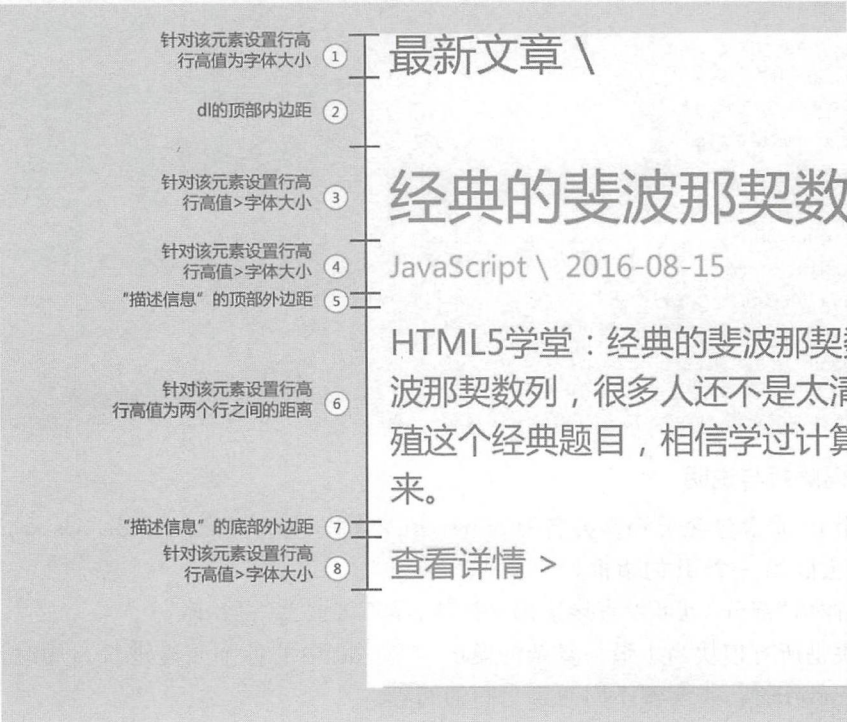


图 19.4 最新文章模块空隙设置方法 2

### 19.1.7 实现内容右侧广告区部分

## 1. 标签选择的基本分析

顶部的教学视频为超链接,单击之后能够发生页面跳转;教学视频下面是一个链接列表,每个“优秀 HTML5 网站链接”都是一个网站名称,可以链接到相应的网站地址;

顶部的教学视频可以直接采用一个 a 标签；下面的各个网站链接，也可以直接使用 a 标签来实现。

额外备注：如果功能需求为“每个网站链接是一个图像”，则应该在 a 标签当中添加 img 标签。

## 2. 标签选择结果

基于上面的基本分析,广告部分的标签选择如下。

[illegible]



```
</div>  
</div>
```

### 3. 模块基本布局样式处理

需要修改类名为 ads 的 div 的样式(加粗部分为进行调整的代码):

```
.ads {  
    float: right;  
    width: 262px;  
    height: 723px;  
    padding-top: 51px;  
}
```

额外添加的样式:

```
.video {  
    display: block;  
    height: 135px;  
    margin-bottom: 2px;  
}  
.friendlink a {  
    display: block;  
    width: 260px;  
    height: 79px;  
    margin-top: 14px;  
    border: 1px solid #dadada;  
}
```

### 4. 结合文本类样式,实现模块开发

与上面的代码相比,没有修改的 CSS 选择器样式并没有书写。

```
.video {  
    display: block;  
    height: 135px;  
    margin-bottom: 2px;  
    background: url('../images/video.jpg') 0 0 no-repeat;  
}  
.friendlink a {  
    display: block;  
    width: 260px;  
    height: 79px;  
    margin-top: 14px;  
    border: 1px solid #dadada;  
    text-align: center;  
    line-height: 79px;  
    font-size: 18px;  
}
```



### 19.1.8 实现 tab 区域标题部分

#### 1. 标签选择的基本分析

设计图当中,共有三个小标题,这三个标题应当对应下面的内容区域,即当前在 tab 内容区域中的三个文章,是属于“最新攻略”部分的。

第一个标题和后面两个标题的样式有所不同,第一个标题处于“激活状态”(即当前 tab 内容区域显示的是该标题对应的内容,其他两个标题为普通状态)。

该区域的功能,是比较经典的 tab 切换效果,通过 JS 进行切换效果的控制,并不需要跳转页面,因此不需要使用 a 标签,使用 span 元素即可。

#### 2. 标签选择结果

基于上面的基本分析,tab 区域的标题部分,标签选择如下。

```
<div class="tabtit">
  <span class="select">最新攻略</span>
  <span>移动端开发攻略</span>
  <span>开发工具攻略</span>
</div>
```

#### 3. 模块基本布局样式处理

```
.tabtit span {
  float: left;
  width: 170px;
  height: 42px;
  background: #fff;
}
.tabtit .select {
  background: #f4f4f4;
}
```

#### 4. 结合文本类样式,实现模块开发

与上面代码相比,没有修改的 CSS 选择器样式并没有书写。

```
.tabtit span {
  float: left;
  width: 170px;
  height: 42px;
  background: #fff;
  text-align: center;
  line-height: 42px;
  cursor: pointer;
}
```

#### 5. 代码解析与说明

为 div 中的所有 span 元素均设置白色的背景,而为处于激活状态的 span 元素设置





#f4f4f4 颜色的背景。通过选择器的优先级,实现样式的覆盖。

虽然在 tab 区域的标题部分并没有超链接,但是为了告知用户能够单击此处,需要为 span 标签设置“光标样式”。

### 19.1.9 实现 tab 区域内容部分

#### 1. 标签选择的基本分析

该部分由三篇文章组成,每篇文章虽然图文并茂,但是如果真的采用 dl 标签,样式操作起来反而会比较复杂,此处可以采用 ul 来包含三篇文章,每个文章使用一个 li 进行处理,文章标题部分使用 h2 标签、图像直接使用 img 标签、段落与查看详情部分使用 div 包含。

单击文章标题或“查看详情”的文字,能够进行页面跳转,因此需要为相应部分添加 a 标签。

#### 2. 标签选择结果

基于上面的基本分析,tab 区域的内容部分,标签选择如下(出于篇幅考虑,如下代码中只书写了一个 li)。

注意,此处将类名为 tabcon 的 div 元素更换为了 ul 元素。

```
<div class = "tabcon - box">
  <ul class = "tabcon">
    <li>
      <h2><a href = "" title = "">使用 Git 多人协作,完成项目开发</a></h2>
      <img src = "images/arcpic.jpg" alt = "" title = "">
      <div>
        <p>HTML5 学堂: 关于 Git 的知识,我们共分成了四个大步骤进行讲解,之前我们提到了 Git 的安装与配置、Git 在本地的使用方法,如何创建 Git 本地仓库与服务器端仓库的关系.</p>
        <a href = "" title = "">查看详情</a>
      </div>
    </li>
  </ul>
</div>
```

#### 3. 模块基本布局样式处理

修改类名为 tabcon 元素的样式:

```
.tabcon {
  width: 990px;
  height: 350px;
  padding-top: 40px;
  margin: 0 auto;
}
```

添加内容标签的样式:

```
.tabcon li {
  float: left;
```



```
        width: 302px;
        margin-left: 42px;
    }
    .tabcon li:first-child {
        margin-left: 0;
    }
    .tabcon h2 {
        height: 56px;
    }
    .tabcon h2 a {
        display: block;
        height: 100%;
    }
    .tabcon img {
        display: block;
        width: 100%;
        height: 136px;
    }
    .tabcon div {
        height: 85px;
        margin-top: 5px;
        padding: 10px 18px;
    }
    .tabcon p {
        height: 54px;
        margin-bottom: 9px;
    }
    .tabcon div a {
        display: block;
        height: 16px;
    }
}
```

#### 4. 结合文本类样式,实现模块开发

与上面代码相比,没有修改的 CSS 选择器样式并没有书写。

```
.tabcon h2 {
    height: 56px;
    text-align: center;
    line-height: 56px;
}
.tabcon div {
    height: 85px;
    margin-top: 5px;
    padding: 10px 18px;
    background: #fbfbfb;
    font-size: 12px;
}
.tabcon p {
    height: 54px;
}
```





```

margin-bottom: 9px;
line-height: 18px;
font-family: 'SimSun';
color: #666;
}
.tabcon div a {
display: block;
height: 16px;
font-family: 'SimSun';
}

```

## 5. 代码解析与说明

该模块与页面中的其他模块相比,采用的字体并不相同,除了微软雅黑之外,在文字描述以及“查看详情”部分,采用的是宋体。

### 19.1.10 实现底部(版权)区域部分

#### 1. 标签选择的基本分析

底部包含两个部分,一个是版权部分,另一部分是底部导航。对于版权部分可以使用 div 实现,对于底部导航是两行文字,因此需要使用 p 标签配合 br 标签、a 标签来实现,也可以采用两个 p 标签(其中一个 p 标签包含各类超链接)来实现。

#### 2. 标签选择结果

基于上面的基本分析,底部版权区域的标签选择如下。

```

<div class="footer-box">
  <div class="footer">
    <div class="copyright">HTML5 学堂网(<a href="" title="">www.h5course.com</a>)
    © 2016 版权所有</div>
    <p>
      <a href="" title="">郑重声明</a>
      <a href="" title="">关于 HTML5 学堂</a>
      <a href="" title="">主创团队</a>
      <a href="" title="">微信公众号</a>
      <a href="" title="">学堂微博</a>
      <a href="" title="">前端人才群</a>
      <a href="" title="">学堂技术交流群</a>
    </p>
    <p>备案号: XXXXXXXXXX 号</p>
  </div>
</div>

```

#### 3. 模块基本布局样式处理

```

.copyright {
padding-top: 32px;
margin-bottom: 16px;
}

```



```

}
.footer p a {
    padding: 0 7px;
    border-left: 1px solid #fff;
}
.footer p a:first-child {
    border-left: none;
}

```

#### 4. 结合文本类样式,实现模块开发

与上面代码相比,没有修改的 CSS 选择器样式并没有书写。

```

.footer {
    text-align: center;
}
.footer, .footer a {
    color: #fff;
}
.copyright {
    padding-top: 32px;
    margin-bottom: 16px;
    font-size: 14px;
    font-weight: bold;
}
.footer p {
    font-size: 12px;
    line-height: 20px;
}

```

### 19.1.11 实现大图部分

#### 1. 标签选择的基本分析

大图部分当中,左右箭头、圆点均处于图片之上,需要使用定位布局。

#### 2. 标签选择结果

基于上面的基本分析,大图部分的标签选择如下。

```

<div class="bigpic">
  <div class="pics">
    <a href="" title="">
      
    </a>
  </div>
  <span class="left"></span>
  <span class="right"></span>
  <ul>
    <li class="now"></li>
    <li></li>
  </ul>
</div>

```





```
</ul>  
</div>
```

### 3. 模块基本布局样式处理

```
.bigpic {  
    position: relative;  
}  
.pics a {  
    float: left;  
    width: 970px;  
    height: 260px;  
}  
.pics a img {  
    display: block;  
    width: 100%;  
    height: 100%;  
}  
.left {  
    position: absolute;  
    top: 50%;  
    left: 24px;  
    width: 20px;  
    height: 36px;  
    margin-top: -18px;  
}  
.right {  
    position: absolute;  
    top: 50%;  
    right: 24px;  
    width: 20px;  
    height: 36px;  
    margin-top: -18px;  
}  
.bigpic ul {  
    position: absolute;  
    left: 50%;  
    right: 0;  
    bottom: 14px;  
    width: 50px;  
    margin-left: -25px;  
}  
.bigpic li {  
    float: left;  
    width: 14px;  
    height: 14px;  
    margin: 0 6px 0 5px;  
}
```





#### 4. 结合文本类样式,实现模块开发

与上面代码相比,没有修改的 CSS 选择器样式并没有书写。

```
.left {
    position: absolute;
    top: 50%;
    left: 24px;
    width: 20px;
    height: 36px;
    margin-top: -18px;
    background: url('../images/picbtns.png') -20px -36px no-repeat;
    cursor: pointer;
}
.right {
    position: absolute;
    top: 50%;
    right: 24px;
    width: 20px;
    height: 36px;
    margin-top: -18px;
    background: url('../images/picbtns.png') -20px 0px no-repeat;
    cursor: pointer;
}
.bigpic li {
    float: left;
    width: 14px;
    height: 14px;
    margin: 0 6px 0 5px;
    background: url('../images/picbtns.png') -3px -72px no-repeat;
    cursor: pointer;
}
.bigpic .now {
    background: url('../images/picbtns.png') -23px -72px no-repeat;
}
```

#### 5. 代码解析与说明

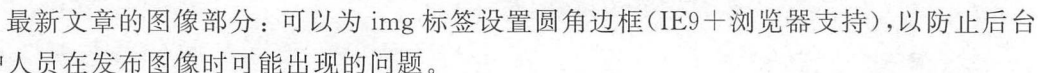
在大图区域当中的左右按键,应当是可以单击的,虽然并没有超链接,但是为了告知用户能够单击此处。需要为标签设置“光标样式”。

### 19.1.12 考虑超出隐藏以及光标移入状态

#### 1. 需要考虑超出的部分

最新文章标题部分:需要处理为超出显示为省略号。

最新文章的介绍(描述信息)部分:需要处理为超出隐藏。

最新文章图像部分:可以为 标签设置圆角边框(IE9+浏览器支持),以防止后台维护人员在发布图像时可能出现的问题。

广告区域的“优秀 HTML5 网站链接”文字部分:需要处理为超出显示为省略号。





tab 内容区的文章标题部分: 需要处理为超出显示为省略号。

tab 内容区的文章介绍部分: 需要处理为超出隐藏。

## 2. 添加超出处理的样式代码

```
.news - con dt img {
    display: block;
    width: 164px;
    height: 164px;
    border: 3px solid #39f;
    border - radius: 50 %;
}
.news - con h2 a {
    overflow: hidden;
    text - overflow: ellipsis;
    word - break: keep - all;
    white - space: nowrap;
    display: block;
    height: 100 %;
    line - height: 40px;
    font - size: 24px;
}
.news - con p {
    overflow: hidden;
    height: 80px;
    margin: 9px 0;
    line - height: 20px;
    font - size: 14px;
    color: #666;
}
.friendlink a {
    overflow: hidden;
    text - overflow: ellipsis;
    word - break: keep - all;
    white - space: nowrap;
    display: block;
    width: 260px;
    height: 79px;
    margin - top: 14px;
    border: 1px solid #dadada;
    text - align: center;
    line - height: 79px;
    font - size: 18px;
}
.tabcon h2 a {
    display: block;
    height: 100 %;
    overflow: hidden;
    text - overflow: ellipsis;
    word - break: keep - all;
```





```

        white - space: nowrap;
    }
    .tabcon p {
        overflow: hidden;
        height: 54px;
        margin - bottom: 9px;
        line - height: 18px;
        font - family: 'SimSun';
        color: # 666;
    }

```

### 3. 需要考虑光标效果的部分

在之前的代码当中,有一些部分已经书写了光标效果,在这里也一一罗列出来,希望读者能够较完整地了解“在网页中需要设置光标移入的部分”。

【此前已设置】所有的 a 标签默认样式,已在重置文件当中进行了设置;

【此前已设置】tab 标题部分,在光标移入时应显示为“小手”状态;

【此前已设置】大图部分的左右键,在光标移入时应显示为“小手”状态;

【需添加】LOGO 部分的大标题在光标移入时应当没有下划线效果;

【需添加】关注 HTML5 学堂的微信图像,在光标移入时应当更换为另一张有颜色的图像;

【需添加】大图部分当中,当光标移入左右按键时,应当更换为颜色不透明的左右键背景图像;

【需添加】光标移入具体的导航项时,导航项的文字不出现下划线,但是背景颜色发生变化。

### 4. 添加光标移入效果的样式代码

```

.header - con a {
    display: block;
    height: 100%;
    padding - left: 93px;
    font - size: 32px;
    background: url('../images/bac.png') 0 0 no - repeat;
    text - decoration: none;
}
.header - con span {
    float: right;
    width: 92px;
    height: 100%;
    line - height: 109px;
    background: url('../images/bac.png') center - 109px no - repeat;
    cursor: pointer;
}
.header - con span: hover {
    background: url('../images/bac.png') center - 218px no - repeat;
}

```





```
.nav a {
    float: left;
    width: 16.66%;
    height: 100%;
    text-align: center;
    line-height: 44px;
    color: #fff;
    text-decoration: none;
}
.nav a: hover {
    background: rgba(255, 255, 255, 0.2);
}
.left: hover {
    background: url('../images/picbtns.png') 0px -36px no-repeat;
}
.right: hover {
    background: url('../images/picbtns.png') 0px 0px no-repeat;
}
```

### 19.1.13 考虑临界值

当前的网页,当浏览器的大小缩小到 990 像素以下时,顶部的蓝色边框、底部(版权区)的蓝色背景以及 tab 内容区的浅灰色背景,只会显示浏览器宽度的大小,拖曳滚动条所查看的其他部分,并没有背景,显示效果很难看。

#### 1. 需要考虑临界值的部分

顶部 7 像素高的蓝色边框;

tab 内容区的浅灰色背景;

底部(版权)区的蓝色背景。

#### 2. 添加临界值控制的样式代码

```
.header {
    min-width: 990px;
    border-top: 7px solid #39f;
}
.tabcon-box {
    min-width: 990px;
    background: #f4f4f4;
}
.footer-box {
    min-width: 990px;
    background: #39f;
}
```

### 19.1.14 代码优化

(1) CSS 文件压缩。



(2) 背景图压缩与合并: 对于前端开发工程师来说, 只需要进行背景图的压缩和合并, 而不需要处理具体的数据图。背景图合并方面, 可以采用多张合并的背景图, 也可以将所有背景图合并到一张图片当中。

(3) 制作 ico 文件: 在制作完成 ico 文件之后, 将 ico 文件引入到 HTML 文件当中。

```
<link rel="shortcut icon" href="images/favicon.ico">
```



## 19.2 移动端网页开发实战

### 19.2.1 功能需求

知识基础: 网页开发实战包含第 13~第 16 章的相关知识, 还会涉及前 12 章中各种知识, 是一个较为综合的网页案例。本节按照网站开发的顺序, 逐步进行代码的书写与讲解。

功能需求: 实现图 19.5 的 HTML 与 CSS 代码开发。网页效果图的 PSD 源文件, 需要用到的 JS 文件, 均可以在“本书的电子资料”中下载。

兼容要求: 移动端设备的各个浏览器, 系统要求兼容 Android4+, iOS7+。

实现方法: 横向使用百分比, 纵向使用 rem, 通过 JS 进行辅助控制。

额外备注: 在该代码实例当中, 所有的代码仅供参考。

移动端项目效果图与功能需求解读说明如图 19.5 所示。

### 19.2.2 整体测绘

#### 1. 与 PC 端测绘的不同点

与 PC 端网页相比, 移动端网页中模块的复杂度较低, 布局工作相对简单。

由于网页各个属性在横向上使用百分比单位进行开发, 是有可能造成微小误差的, 这种由百分比小数位数导致的微小误差, 是允许发生的。

移动端的整体测绘工作, 除了和 PC 端相同的内容之外, 还需确定设备支持的最大分辨率以及最小字体。

#### 2. 整体测绘结果

整体的测绘结果如下。

(1) 设计图尺寸为 1080 像素, 即能够兼容的最大分辨率大小为 1080 像素, 当屏幕宽度大于 1080 像素时, 网页按照宽度 1080 像素进行渲染; 当屏幕宽度小于 1080 像素时, 网页按照屏幕宽度进行渲染, 实现自适应;

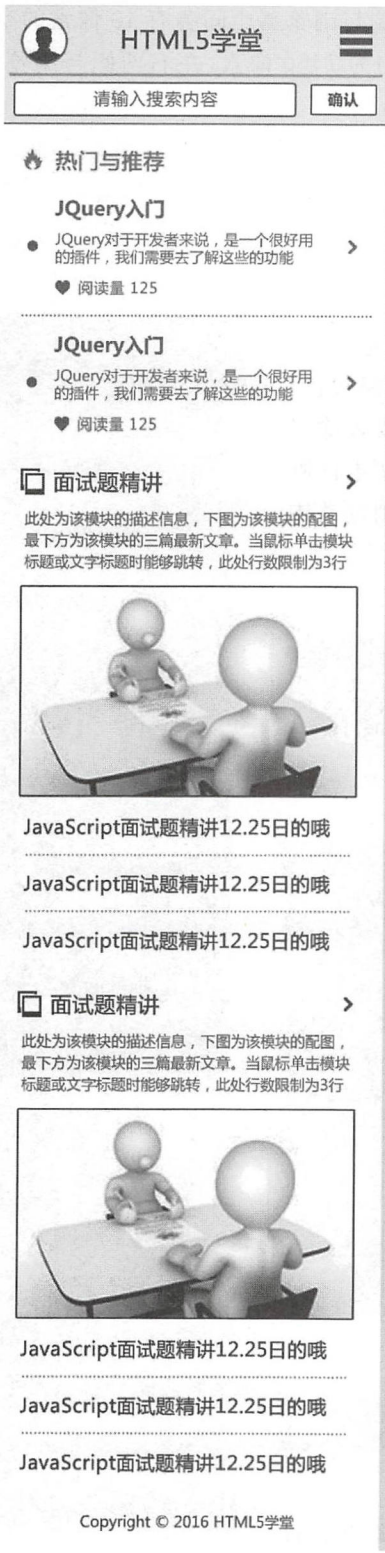
(2) 屏幕最小宽度分辨率 320 像素, 不需要考虑 320 像素以下的屏幕分辨率;

(3) 设计图中的最小字体大小为 42 像素, 符合移动端对字体的基本要求;

(4) 网页可以简单地划分为 4 个部分, 分别是头部、热门与推荐、不同类别的具体文章模块、版权区。

(5) 头部高度为 330 像素, 底部有 4 像素的深灰色边线, 颜色为 #666;





- 1 头部的标题、搜索框，应当为固定定位位于具体内容元素的上面。
- 2 第一部分为热门与推荐，放置两篇文章单击文章之后应当能够跳转到内容页
- 3 第三部分为具体的模块，在PSD图当中出现了两次“面试题精讲”的模块，实际给开发工程师传达的功能需求为：此处呈现具体的模块，每个模块均由模块名称、模块简介、模块配图组成，单击模块标题，能够跳转到相应的模块列表页。每个模块底部有该模块最新的三篇文章，单击标题能够跳转到相应的文章。
- 4 第四部分为版权区域

图 19.5 移动端网页效果图与功能需求解读

(6) 热门与推荐部分,高度为 920 像素,左右两侧与浏览器之间各有 40 像素的空隙;

(7) 不同类别的具体文章模块,每个模块高度均为 1460 像素,左右两侧与浏览器之间各有 40 像素的空隙;

(8) 底部版权部分,高度 145 像素,水平居中。

### 3. 测绘示意图

整体测绘图如图 19.6 所示。

### 4. 调整重置文件

在该页面的设计图当中,大部分的链接颜色均为 #666 的深灰色,部分没有链接的文字颜色为 #303030 的深灰色,字体均为微软雅黑,最小字体为 42px。此外,还需要去除光标移入效果;去除单击链接时的高亮部分,浏览器文本框默认高光样式,iPhone、iPad 按钮的默认样式,IE10+浏览器中文本框的叉号。

需要为重置文件修改如下部分:

```
html {color: # 303030; background: # FFF; font - family: '
Microsoft YaHei', sans - serif, Arial;}
a{text - decoration:none; color: # 666; outline:none;}
```

需要为重置文件添加如下部分:

```
a {
    -webkit - tap - highlight - color: rgba(255,0,0, 0);
}
input:focus{
    -webkit - tap - highlight - color: rgba(0,0,0,0);
    -webkit - user - modify: read - write - plaintext - only;
}
input[ type = "button"], input[ type = "submit"], input[ type =
"reset"] {
    -webkit - appearance: none;
}
input::-ms-clear {
    display: none;
}
```

需要为重置文件删除如下部分:

```
a:hover{text - decoration:underline;}
```

### 5. 添加 meta 信息

通过 meta 标签进行视口的控制,同时需要去除设备自动识别电话、邮箱的功能。

需要添加如下 meta 信息:

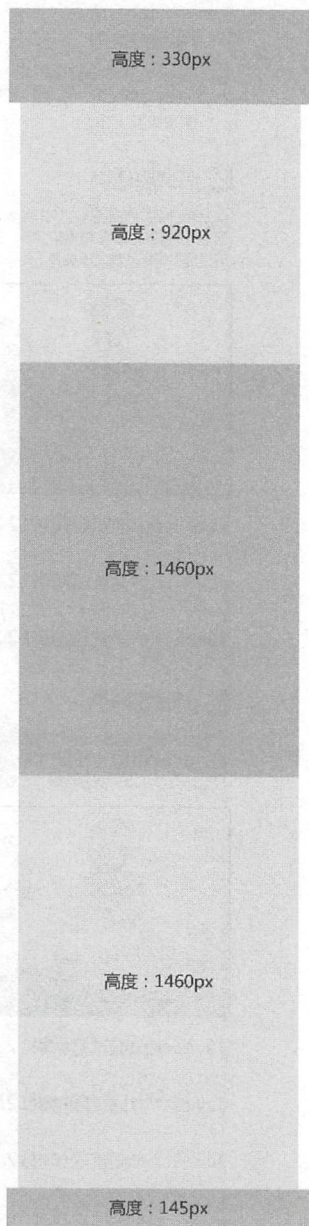


图 19.6 移动端网页开发整体测绘图



```
<meta name="viewport" content="width=device-width,user-scalable=no">
<meta content="telephone=no" name="format-detection">
<meta content="email=no" name="format-detection">
```

### 19.2.3 实现整体布局

#### 1. 整体布局的相关代码

结构代码：

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 布局之路 - 移动端网页</title>
  <link rel="stylesheet" href="css/reset.css">
  <link rel="stylesheet" href="css/index.css">
  <meta name="viewport" content="width=device-width,user-scalable=no">
  <meta content="telephone=no" name="format-detection">
  <meta content="email=no" name="format-detection">
</head>
<body>
  <header></header>
  <div class="box">
    <div class="hots"></div>
    <section class="model"></section>
    <section class="model"></section>
    <footer></footer>
  </div>
</body>
</html>
```

517

样式代码(index.css),左侧代码以 px 为单位,右侧代码以百分比与 rem 为单位。

```
@charset 'utf-8';
html {
  font-size: 42px;
}
header {
  position: fixed;
  top: 0;
  left: 0;
  z-index: 9;
  width: 100%;
  max-width: 1080px;
  height: 330px;
  box-shadow: 0 4px #666;
}
```

```
@charset 'utf-8';
html {
  font-size: 42px;
}
header {
  position: fixed;
  top: 0;
  left: 0;
  z-index: 9;
  width: 100%;
  max-width: 1080px;
  height: 7.86rem;
  box-shadow: 0 0.09rem #666;
}
```

```
.box {
    max-width: 1080px;
    padding-top: 334px;
}
.hots {
    padding: 0 40px;
    height: 920px;
}
.model {
    padding: 0 40px;
    height: 1460px;
}
footer {
    height: 145px;
}
/* 如下代码用于标注标签,后期统一删除 */
header, .model:nth-of-type(1), footer {
    background: #ccc;
}
.hots, .model:nth-of-type(2) {
    background: #e9e9e9;
}
```

```
.box {
    max-width: 1080px;
    padding-top: 7.86rem;
}
.hots {
    padding: 0 3.7%;
    height: 21.9rem;
}
.model {
    padding: 0 3.7%;
    height: 34.76rem;
}
footer {
    height: 3.45rem;
}
/* 如下代码用于标注标签,后期统一删除 */
header, .model:nth-of-type(1), footer {
    background: #ccc;
}
.hots, .model:nth-of-type(2) {
    background: #e9e9e9;
}
```

## 2. 代码解析与说明

在开发时,首先应当引入 reset.css 文件,以防止浏览器默认样式出现问题;

<header>标签需要一直显示在网页的顶部,并不随着内容的滚动而运动,因此需要使用固定定位,由于<header>标签进行了固定定位,脱离了文档流,就使得类名为 box 的 div,从 body 的左上方开始显示,为了防止内容被遮盖,需要为该 div 设置顶部的 padding 值,值的大小等于<header>标签的具体高度;

为顶部元素设置 z-index,目的在于防止页面中其他定位元素会覆盖该元素;

对于“为了标识布局块”而为布局块设置的背景颜色。建议将此类代码统一书写于所有样式代码的底部,方便后期删除。

### 19.2.4 添加 JS 控制,实现多分辨率自适应

#### 1. 引入 JS 文件

在 HTML 文件的头部引入 zepto.js 和 changeFont.js(在第 14 章当中有详细介绍)。

```
<head>
    <meta charset = "UTF-8">
    <title>HTML5 布局之路 - 移动端网页</title>
    <link rel = "stylesheet" href = "css/reset.css">
    <link rel = "stylesheet" href = "css/index.css">
    <meta name = "viewport" content = "width = device-width,user-scalable = no">
    <meta content = "telephone = no" name = "format-detection">
```



```
<meta content="email=no" name="format-detection">
<script src="js/zepto.js"></script>
<script src="js/changeFont.js"></script>
</head>
```

## 2. 修改 changeFont.js 的内容

修改 changeFont.js 中的字体以及最大分辨率的数值,具体设置如下。

```
$(function(){
    $(window).resize(infinite);
    function infinite() {
        var htmlWidth = $('html').width();
        if (htmlWidth >= 1080) {
            $('html').css({
                "font-size": "42px"
            });
        } else {
            $('html').css({
                "font-size": 42 / 1080 * htmlWidth + "px"
            });
        }
    }
    infinite();
});
```

519

## 3. 页面测试

可以使用手机真机,也可以拖曳浏览器窗口,如果网页能够正常适应(按照等比例大小变化),则说明没有问题。请每书写一个模块的代码就立即进行测试,及时排查问题和错误。

### 19.2.5 实现热门与推荐部分

#### 1. 标签选择的基本分析

热门与推荐,是一个包含两篇文章的列表,非图文的列表部分可以选用 ul 来实现。对于每个列表项,均包含文章标题(h2)、描述信息(p)以及阅读量(div),左侧的圆点以及右侧的“>”符号,可以通过伪元素来实现。

在移动端,出于单击区域的考虑,通常在文章列表项的任意位置单击,均能够实现跳转,因此,此处使用 a 标签包含列表项中的各个标签。

#### 2. 标签选择结果

基于上面的基本分析,“热门与推荐”部分的标签选择如下。

```
<div class="hots">
    <div>热门与推荐</div>
    <ul>
        <li>
            <a href="" title="">
                <h2>jQuery 入门</h2>
```

```

        <p>jQuery 对于开发者来说,是一个很好用的插件,需要去了解这些功能</p>
        <div>阅读量 125 </div>
    </a>
</li>
<li>
    <a href = "" title = "">
        <h2>jQuery 入门</h2>
        <p>jQuery 对于开发者来说,是一个很好用的插件,需要去了解这些功能</p>
        <div>阅读量 125 </div>
    </a>
</li>
</ul>
</div>

```

### 3. 模块样式处理

左侧代码以 px 为单位,右侧代码以百分比与 rem 为单位。

```

.hots > div {
    height: 122px;
    padding: 42px 0 0 105px;
    font-size: 60px;
    color: #f62828;
    line-height: 122px;
    background: url('../images/hots.png')
    15px 68px no-repeat;
    background-size: 55px 60px;
}
.hots li {
    border-bottom: 6px dotted #666;
}
.hots li:last-child {
    border-bottom: 6px dotted #fff;
}
.hots li a {
    position: relative;
    display: block;
    height: 100%;
    padding: 0 156px 0 108px;
}
.hots li a:before {
    content: "\200B";
    position: absolute;
    top: 162px;
    left: 28px;
    width: 30px;
    height: 30px;
    background: #666;
    border-radius: 50%;
}

```

```

.hots > div {
    height: 2.9rem;
    padding: 1rem 0 0 10.5%;
    font-size: 1.43rem;
    color: #f62828;
    line-height: 2.9rem;
    background: url('../images/hots.png')
    0.36rem 1.62rem no-repeat;
    background-size: 1.31rem 1.43rem;
    /* 父级宽度为 1000 像素 */
}
.hots li {
    border-bottom: 0.14rem dotted #666;
}
.hots li:last-child {
    border-bottom: 0.14rem dotted #fff;
}
.hots li a {
    position: relative;
    display: block;
    height: 100%;
    padding: 0 15.6% 0 10.8%;
    /* 父级宽度为 1000 像素 */
}
.hots li a:before {
    content: "\200B";
    position: absolute;
    top: 3.86rem;
    left: 2.8%;
    width: 3%;
    height: 0.71rem;
    background: #666;
}

```



```

.hots li a:after {
    content: "\200B";
    position: absolute;
    top: 0;
    right: 0;
    width: 156px;
    height: 100%;
    background: url('../images/more.png')
90px center no-repeat;
    background-size: 25px 40px;
}
.hots h2 {
    overflow: hidden;
    text-overflow: ellipsis;
    word-break: keep-all;
    white-space: nowrap;
    height: 112px;
    padding-top: 12px;
    font-size: 56px;
    font-weight: bold;
    line-height: 112px;
}
.hots p {
    overflow: hidden;
    height: 100px;
    line-height: 50px;
}
.hots li div {
    height: 42px;
    margin: 42px 0 58px;
    padding-left: 67px;
    line-height: 42px;
    background: url('../images/love.png') 0 0
no-repeat;
    background-size: 42px 42px;
}

border-radius: 50%;
/* 父级宽度为 1000 像素 */
}
.hots li a:after {
    content: "\200B";
    position: absolute;
    top: 0;
    right: 0;
    width: 15.6%;
    height: 100%;
    background: url('../images/more.png') 2.
14rem center no-repeat;
    background-size: 0.6rem 0.95rem;
}
.hots h2 {
    overflow: hidden;
    text-overflow: ellipsis;
    word-break: keep-all;
    white-space: nowrap;
    height: 2.67rem;
    padding-top: 0.29rem;
    font-size: 1.33rem;
    font-weight: bold;
    line-height: 2.67rem;
}
.hots p {
    overflow: hidden;
    height: 2.38rem;
    line-height: 1.19rem;
}
.hots li div {
    height: 1rem;
    margin: 1rem 0 1.38rem;
    padding-left: 9.1%;
    line-height: 1rem;
    background: url('../images/love.png') 0
0 no-repeat;
    background-size: 1rem 1rem;
/* 父级宽度为 736 像素 */
}

```

#### 4. 代码解析与说明

在样式代码书写的过程中,可以按照代码的作用分为三大部分。弄清楚这三部分,能够让书写代码的思路变得更加清晰。

##### 1) PC 端基本样式

首先可以使用绝对度量单位,完成基本样式的书写,此时,在 1080 像素屏幕大小下,网页应当正常显示。此处的基本样式,指的是布局以及文本样式处理,包括 CSS 的自身属性、

显示属性、文本属性以及 CSS3 圆角、阴影等命令。

### 2) 可触区与超出控制

当书写完成基本的结构样式之后,要进行超出隐藏的控制以及 a 标签、img 标签的设置(如上左侧代码中的加粗部分)。

### 3) 移动端调整

最后一部分代码,是将“PC 端代码”转换为“移动端代码”,让网页能够适配不同分辨率的设备。该类代码主要包括 background-size 的设置、绝对度量单位向相对度量单位的转换(如上右侧代码中的加粗部分)。

## 19.2.6 实现具体文章模块部分

### 1. 标签选择的基本分析

在每个具体文章模块当中,由模块名称、模块描述、模块配图以及该模块最新的三篇文章组成。模块标题可以使用 h2 标签,模块的描述信息可以使用 p 标签,模块的配图需要使用 img 标签。在单击模块标题时,应当能够跳转到相应类型的文章列表页;在单击具体三篇文章的标题时,应当能够跳转到相应文章页。

### 2. 标签选择结果

基于上面的基本分析,“具体文章模块”部分的标签选择如下。

```
<section class="model">
  <h2><a href="" title="">面试题精讲</a></h2>
  <p>此处为该模块的描述信息,下图为该模块的配图,最下方为该模块的三篇最新文章.当鼠标单击模块标题或文字标题时能够跳转,此处行数限制为 3 行</p>
  <div></div>
  <ol>
    <li><a href="" title="">JavaScript 面试题精讲 12.25 日的哦</a></li>
    <li><a href="" title="">JavaScript 面试题精讲 12.24 日的哦</a></li>
    <li><a href="" title="">JavaScript 面试题精讲 12.25 日的哦</a></li>
  </ol>
</section>
```

### 3. 模块样式处理

左侧代码以 px 为单位,右侧代码以百分比与 rem 为单位。

```
u.model h2 a {
  position: relative;
  display: block;
  height: 150px;
  padding: 13px 156px 0 108px;
  line-height: 150px;
  font-size: 60px;
  font-weight: bold;
  color: #333;
}
```

```
.model h2 a {
  position: relative;
  display: block;
  height: 3.57rem;
  padding: 0.31rem 15.6% 0 10.8%;
  line-height: 3.57rem;
  font-size: 1.43rem;
  font-weight: bold;
  color: #333;
}
```



```

.model h2 a:before {
    content: "\200B";
    position: absolute;
    top: 50px;
    left: 10px;
    width: 66px;
    height: 72px;
    background: url('../images/model.png') 0
0 no-repeat;
    background-size: 66px 72px;
}
.model h2 a:after {
    content: "\200B";
    position: absolute;
    top: 0;
    right: 0;
    width: 156px;
    height: 100%;
    background: url('../images/more.png')
90px 72px no-repeat;
    background-size: 25px 40px;
}
.model p {
    overflow: hidden;
    height: 180px;
    margin-top: 6px;
    padding: 0 44px 0 22px;
    line-height: 60px;
    color: #666;
}
.model div {
    height: 580px;
    margin: 45px 36px 10px 14px;
    border: 4px solid #000;
}
.model img {
    display: block;
    width: 100%;
    height: 100%;
}
.model ol {
    padding: 0 50px 0 33px;
}
.model li {
    border-bottom: 6px dotted #999;
}
.model li:last-child {
    border: none;
}

```

```

.model h2 a:before {
    content: "\200B";
    position: absolute;
    top: 1.19rem;
    left: 1%;
    width: 6.6%;
    height: 1.71rem;
    background: url('../images/model.png') 0
0 no-repeat;
    background-size: 1.57rem 1.71rem;
}
.model h2 a:after {
    content: "\200B";
    position: absolute;
    top: 0;
    right: 0;
    width: 3.71rem;
    height: 100%;
    background: url('../images/more.png') 2.
14rem 1.71rem no-repeat;
    background-size: 0.6rem 0.95rem;
}
.model p {
    overflow: hidden;
    height: 4.29rem;
    margin-top: 0.14rem;
    padding: 0 4.4% 0 2.2%;
    line-height: 1.43rem;
    color: #666;
}
.model div {
    height: 13.81rem;
    margin: 1.07rem 3.6% 0.24rem 1.4%;
    border: 0.1rem solid #000;
}
.model img {
    display: block;
    width: 100%;
    height: 100%;
}
.model ol {
    padding: 0 5% 0 3.3%;
}
.model li {
    border-bottom: 0.14rem dotted #999;
}
.model li:last-child {
    border: none;
}

```



```
.model li a {
    overflow: hidden;
    text-overflow: ellipsis;
    word-break: keep-all;
    white-space: nowrap;
    display: block;
    height: 152px;
    line-height: 152px;
}
```

```
.model li a {
    overflow: hidden;
    text-overflow: ellipsis;
    word-break: keep-all;
    white-space: nowrap;
    display: block;
    height: 3.62rem;
    line-height: 3.62rem;
}
```

#### 4. 代码解析与说明

依旧按照“PC 端基本样式”→“可触区与超出控制”→“移动端调整”三步进行操作。

### 19.2.7 实现顶部区域

#### 1. 顶部导航部分

顶部区域的上半部分,在第 14 章当中已经进行了详细的分析与讲解,在此不再详细讲解,直接给出完成版代码。(由于需要考虑整体结构和样式,因此在这里并没有直接挪用 14 章中的源代码,而是在其基础之上进行了微调。)

结构代码:

```
<header>
  <nav>
    <span></span>
    <h1>HTML5 学堂</h1>
    <strong></strong>
  </nav>
</header>
```

样式代码,左侧代码以 px 为单位,右侧代码以百分比与 rem 为单位。

```
header {
    background: #eee;
}
nav {
    width: 1052px;
    height: 195px;
    margin: 0 auto;
    box-shadow: inset 0 -5px #a6a6a6;
}
nav span {
    float: left;
    width: 121px;
    height: 121px;
    padding: 35px 38px;
}
```

```
header {
    background: #eee;
}
nav {
    width: 97.33%;
    height: 4.64rem;
    margin: 0 auto;
    box-shadow: inset 0 -0.12rem #a6a6a6;
}
nav span {
    float: left;
    width: 11.2%;
    height: 2.88rem;
    padding: 0.83rem 3.52%;
}
```



```

nav img {
    display: block;
    width: 100%;
    height: 100%;
    box-shadow: 0 0 0 6px #727272;
    border-radius: 50%;
}
nav h1 {
    float: left;
    width: 658px;
    font-size: 72px;
    text-align: center;
    line-height: 195px;
}
nav strong {
    float: right;
    width: 197px;
    height: 191px;
    background: url('../images/menu.png') 0
0 no-repeat;
    background-size: 100%;
}

```

```

nav img {
    display: block;
    width: 100%;
    height: 100%;
    box-shadow: 0 0 0 0.14rem #727272;
    border-radius: 50%;
}
nav h1 {
    float: left;
    width: 60.93%;
    font-size: 1.71rem;
    text-align: center;
    line-height: 4.64rem;
}
nav strong {
    float: right;
    width: 18.24%;
    height: 4.55rem;
    background: url('../images/menu.png') 0
0 no-repeat;
    background-size: 100%;
}

```

## 2. 顶部搜索部分——标签选择的基本分析

搜索部分由两个部分组成,分别是文本输入框和提交按钮,用于让用户在网站中实现快速搜索功能。由于需要将用户输入的数据提交到后台服务器,进行数据查找,因此,需要使用 form 表单元素,而文本输入框和按钮,分别使用 text 类型和 submit 类型的 input 元素来实现。

## 3. 顶部搜索部分——标签选择结果

基于上面的基本分析,“搜索”部分的标签选择如下。

```

<form class="search" action="" method="">
    <div>
        <input type="text" name="" id="" placeholder="请输入搜索内容">
        <input type="submit" value="确认" name="" id="">
    </div>
</form>

```

## 4. 顶部搜索部分——模块样式处理

左侧代码以 px 为单位,右侧代码以百分比与 rem 为单位。



```

.search div {
    padding: 16px 31px 19px 27px;
}
.search input[type = "text"] {
    float: left;
    width: 788px;
    height: 89px;
    border: 4px solid #303030;
    border-radius: 10px;
    text-align: center;
    line-height: 89px;
    font-size: 48px;
    color: #303030;
}
.search input[type = "submit"] {
    float: right;
    width: 179px;
    height: 80px;
    margin-top: 4px;
    padding: 0;
    border: 4px solid #303030;
    border-radius: 5px;
    background: none;
    text-align: center;
    line-height: 72px;
    font-weight: bold;
    color: #303030;
}
.search input[type = "text"]:-moz-placeholder {
    color: #555;
}
.search input[type = "text"]::-moz-placeholder {
    color: #000;
}
.search input[type = "text"]:-ms-input-placeholder {
    color: #555;
}
.search input[type = "text"]::-webkit-input-placeholder {
    color: #555;
}

```

```

.search div {
    padding: 0.38rem 2.87% 0.45rem 2.5%;
}
.search input[type = "text"] {
    float: left;
    width: 77.1%;
    height: 2.12rem;
    border: 0.1rem solid #303030;
    border-radius: 0.24rem;
    text-align: center;
    line-height: 2.12rem;
    font-size: 1.14rem;
    color: #303030;
}
.search input[type = "submit"] {
    float: right;
    width: 17.51%;
    height: 1.9rem;
    margin-top: 0.1rem;
    padding: 0;
    border: 0.1rem solid #303030;
    border-radius: 0.12rem;
    background: none;
    text-align: center;
    line-height: 1.71rem;
    font-weight: bold;
    color: #303030;
}
.search input[type = "text"]:-moz-placeholder {
    color: #555;
}
.search input[type = "text"]::-moz-placeholder {
    color: #000;
}
.search input[type = "text"]:-ms-input-placeholder {
    color: #555;
}
.search input[type = "text"]::-webkit-input-placeholder {
    color: #555;
}

```

## 5. 顶部搜索部分——代码解析与说明

placeholder 的字体样式,在火狐下虽然也能够设置,但是颜色会不一致,即便为火狐设置的是黑色(#000),依旧会按照#767676的灰色进行处理。



## 19.2.8 实现底部版权部分

### 1. 标签选择的基本分析与标签选择结果

版权部分比较简单,只有一行文字,且没有任何链接。因此,并不需要添加额外标签,只需要在原有的< footer ></ footer >当中添加内容即可。

```
< footer > Copyright © 2016 HTML5 学堂</ footer >
```

### 2. 模块样式处理

左侧代码以 px 为单位,右侧代码以百分比与 rem 为单位。

```
footer {  
    height: 145px;  
    text-align: center;  
    line-height: 145px;  
}
```

```
footer {  
    height: 3.45rem;  
    text-align: center;  
    line-height: 3.45rem;  
}
```

527

## 19.2.9 代码优化

(1) 不需要进行新结构标签的兼容处理: 由于移动端的各种设备,不同系统中的各类浏览器,均对 HTML5 有着比较良好的支持,因此并不需要使用 JS 进行新结构标签的兼容处理。对于圆角边框、阴影、新增背景属性、新增选择器等,均可以正常使用。

(2) CSS 文件压缩、背景图压缩、制作 ico 文件: 与此前 PC 端网页中讲解的相同,在此不再讲解。

(3) “热门与推荐的列表区域”调整为内容撑开高度(去掉, hots{ } 中的高度设置即可),好处在于有利于后期维护,如果改变需求,希望能够显示三条数据,则前端不需要进行任何更改。此外,具体模块部分的每个模块的高度也可以不做限制,由内容撑开高度(去掉, model{ } 代码中的高度设置即可)。

# 第20章

## 附 录



### 20.1 HTML5 发展史

#### 20.1.1 萌芽

HTML 是 HyperText Markup Language 的简写,表示超文本标记语言。标记语言出现的很早,比 1972 年 C 语言推出还要早一些,在 1969 年出现了 GML 通用标记语言,后来在 1985 年,IBM 人员创立了 SGML,但是后来因为使用成本太高,在比较长的一段时间内并没有得到推广。

HTML 的正式诞生是在 1989 年,刚刚诞生的 HTML,规范很少、功能简单,在发展过程中不断地添加着新标签,这种现象也就直接导致了后来整个 HTML 体系变得很臃肿,兼容性差。

#### 20.1.2 第一次浏览器大战

1994 年,HTML2 版本诞生,第一次互联网大战也随之爆发。1994 年,网景公司推出了 Netscape0.9 浏览器,迅速占领了 90% 的市场份额,在浏览器发布 6 个月之后,网页成为互联网最主要的应用之一。与网景公司相抗衡的是微软公司,在 1995 年 8 月,微软推出了首个浏览器 IE1.0。

1997 年,HTML4 标准出台。微软的浏览器成为第一大浏览器厂商,市场份额达到 49%,超越了网景的 46%。短短三年期间浏览器市场份额发生巨变的原因,除了浏览器在费用方面的差别之外(IE 浏览器免费,而网景浏览器收费),还有一些其他的原因(如:网景主要的平台是服务器厂商;IE 浏览器直接绑定于微软的 Windows 系统当中等)。

1998 年,网景为了挽回市场,将公司的绝大多数产品均转为“免费”,同时设立了开源组织 Mozilla Organization。但是这些调整并没有能够挽救网景,1998 年,网景通讯公司被美国线上买下,2003 年网景公司解散,Mozilla Organization 改组为 Mozilla 基金会(也是火狐浏览器的前身),自此,第一次浏览器大战告终。

#### 20.1.3 第二次浏览器大战

2001 年到 2009 年,是第二次浏览器大战,Mozilla 组织研发出了火狐浏览器,火狐浏览器开始与 IE 浏览器争夺市场份额。由于微软公司此前的自满,未及时更新推出 IE 浏览器





的新版本和新功能,而在第一次大战当中的后遗症也开始逐渐显现(由于当时追求功能而忽略了安全性的问题)。安全性的问题逐渐引起了使用方的高度重视,欧洲数个国家宣布放弃IE,改用火狐浏览器。

### 20.1.4 第三次浏览器大战

2008年,HTML5 第一个版本草案推出,谷歌(Google)公司先后推出各平台下对HTML5 支持较好的 Chrome 浏览器,HTML5 版本推出,各个浏览器都面对着全新的网站开发标准以及移动端的挑战。

由于谷歌、苹果浏览器对于 HTML5 最新功能的完善支持,在移动端的推动之下迅速崛起;2012 年微软推出 IE9 版本浏览器,开始支持 HTML5 等新技术。在 2012 年,IE、火狐、谷歌呈三足鼎立之势。2013 年年底,IE10、IE11 的迅速推进,Windows 8 系统的普及,让 IE 浏览器的市场占有率终于停止了下降。

浏览器发展史如图 20.1 所示。

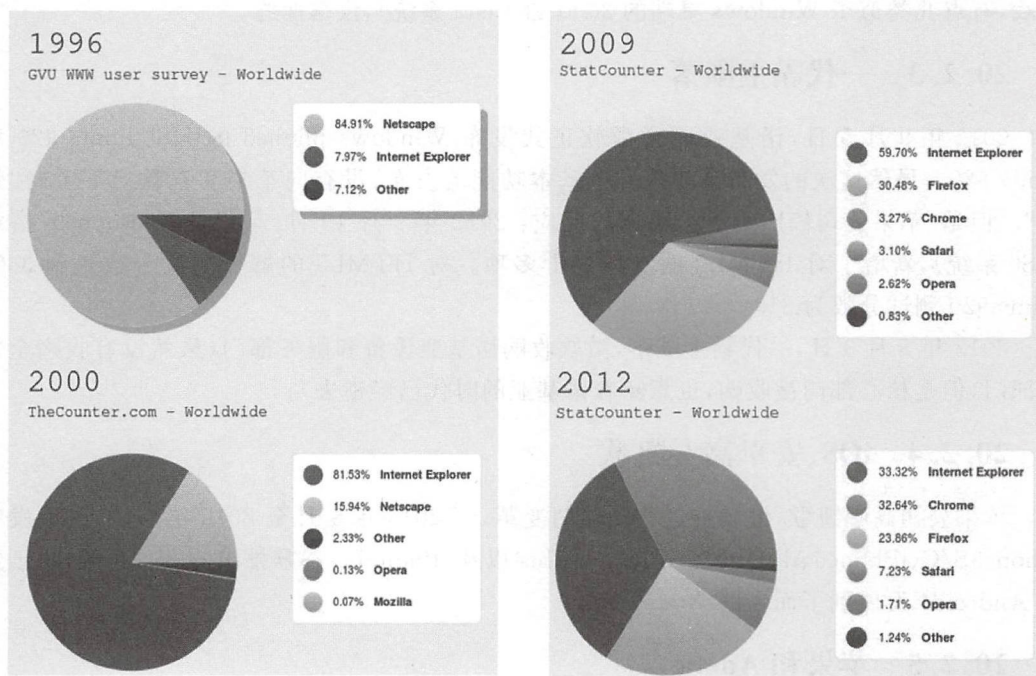


图 20.1 浏览器发展史



## 20.2 手机端操作系统发展史

### 20.2.1 诺基亚的世界开始动摇

2007 年以前是诺基亚的天下。2007 年之前还没有智能手机,那时候诺基亚是手机的王者,各种型号各种类型,有着不可震撼的地位。

2007 年 1 月,苹果推出第一款 iPhone,同年 6 月,苹果发布 Safari 浏览器。2008 年,谷



歌在各个平台推出 Chrome 浏览器,发布第一个 HTML5 版本草案,2008 年 9 月,推出安卓 1.0 版本系统。诺基亚在当时,并不看好苹果谷歌的产品,诺基亚认为:在一种失去触觉感受的状态下,苹果是不可能做成功的。但是诺基亚忽略了一个很重要的因素,虽然没有手指按下的感觉,但是功能更强,屏幕更大,让用户更加舒服。

### 20.2.2 微软、苹果、谷歌之战

2011 年 9 月 25 日,微软发布 Windows phone 7.5(芒果)。搭载 IE9 浏览器,渲染 HTML5 性能测试页面,芒果达到 25 帧,同期 Android 10 帧,iOS 个位数,以此来与 iOS4 系统和 Android2 版本系统相比较。

苹果公司和谷歌公司在微软发布之后迅速反应,2011 年 10 月 12 日,iOS5 版本发布,渲染同组 HTML5 性能测试页面,达到 35 帧;2011 年 10 月 19 日,Android 4 版本发布。新的 HTML5 解析引擎提速 35%~500%。由于苹果与谷歌的快速反应以及系统升级,使得微软公司的芒果手机面临尴尬局面,最终不得不停止掉了所有的支持(不为其开发应用程序和游戏,有点儿类似于 Windows 桌端的 2000 和 Vista 系统),没落而终。

### 20.2.3 一代霸主陨落

2012 年 9 月 5 日,诺基亚联合微软正式发布 Windows phone8 的手机 lumia 920 和 lumia 820。虽然这次的发布会就是一个基本功能的发布,没有向苹果和谷歌扔下重磅“炸弹”。但是,苹果公司依旧给予了快速的反应:2012 年 9 月 13 日,苹果发布 iPhone5(搭载 iOS6 系统),新增了对 HTML5 的支持二十多项。对 HTML5 的解析测试分数达到 360,Lumia920 测试分数为 319。

2013 年 9 月 3 日,一代霸主陨落,微软收购诺基亚设备和服务部门(虽然没有收购全部的部门,但是核心部门被收购,也意味着诺基亚的时代已经逝去)。

### 20.2.4 iOS、安卓高奏凯歌

苹果公司高唱凯歌,继续推进移动端的变革,于 2013 年 9 月至 2016 年 9 月,先后发布 iPhone5S/C、iPhone6、iPhone6S、iPhone6 Plus 以及 iPhone7。谷歌推出的安卓系统,也已经从 Android4 发展到了而今的 Android 7。

### 20.2.5 苹果和 Adobe

2010 年,苹果大战 Adobe 公司。iPhone4 出世,4 月份,乔布斯宣布苹果移动设备将不支持 Flash,仅支持 HTML5 的视音频,自此掀起了苹果与 Adobe 的一片骂战,也掀起了 Flash 和 HTML5 开发工作者的一场战争。

2011 年 10 月,Adobe 进行产品 DW 的整合,将 phonegap 整合到 DW 当中。

2011 年 11 月 10 日,Adobe 公司宣布停止开发移动版 Flash 视频播放,转而加大对 HTML5 的投入。同日,Adobe 再次宣布放弃为电视开发 Flash 插件,并且不再支持通过 Flash 开发移动设备上的应用,转为 HTML5 和 AIR 的应用。

自此,这场 Flash 与 HTML5 的大战,以 Flash 失败而告终。







## 20.3 HTML 的各种布局

### 20.3.1 表格布局

而今采用表格布局的网页越来越少,有一部分大学、中学的学校官网还没有升级,有可能采用的是此类布局,进入网站之后按 F12 键即可查看结构。

由于布局时不会出现错乱,表格曾经一度被开发者用于布局。随着对 SEO 以及代码维护性、扩展性的要求,表格逐渐走下神台。

### 20.3.2 DIV+CSS

替代表格布局而存在的,就是 DIV+CSS 布局,更准确地说,其实应该是 HTML+CSS。这种布局方式,主要通过 div 元素来进行页面元素的排版,通过 CSS 来定义 div 的具体样式。(当然并不意味着网页中所有的元素都使用 div。)

这种布局方式在 SEO、代码操作性、维护性、代码量等方面都优于 table 布局。接下来的几种布局方式,其实都属于 DIV+CSS(HTML+CSS)布局中的子类,因为具有一定的时代特点,因此单独拿出来介绍一下。

### 20.3.3 960 栅格

960 栅格化系统是由 Nathan Smith 创造的,具体来讲,就是基于 960 像素宽度,通过划分具有规律的分割,来提高网页开发效率。

960 栅格化系统(或者说适应型 CSS 构架)早期主要用来进行快速原型制作,减少重复单调的工作,但是后来,它在网页设计开发项目中逐渐扮演非常重要的角色了。它将为网页的设计提供一个坚实的坐标基础。

设计师使用 960 栅格进行设计,而开发工程师,则是利用 CSS 的组合式进行样式的书写。(类的组合可详见第 4.3 节。)

这种书写方式主要针对的是 1024 的纯平显示器时代,于是就出现了如图 20.2 中的各种“标准化”的布局方式(而今的各个博客网站的用户博客当中,还能够看到这类布局)。

随着分辨率的发展,这种布局方式逐渐被摒弃掉,一方面是由于 960 栅格的布局方式在维护上并不是太方便,另一方面,也是最重要的是:当屏幕的分辨率从原有的 1024 越来越大,且没有规律可循时,设计师并不再一板一眼地按照 960 栅格进行网页设计。

### 20.3.4 经典的三栏布局——双飞翼

双飞翼布局是一种灵活的布局,始于淘宝 UED,应该是玉伯提出的。如果把三栏布局比作一只大鸟,可以把 main 看成是鸟的身体,sub 和 extra 则是鸟的翅膀。这个布局的实现思路是,先把最重要的身体部分放好,然后再将翅膀移动到适当的地方。

随着网页设计图的复杂化,这种比较经典的三栏布局,在网页当中越来越少,使用的也就越来越少了。



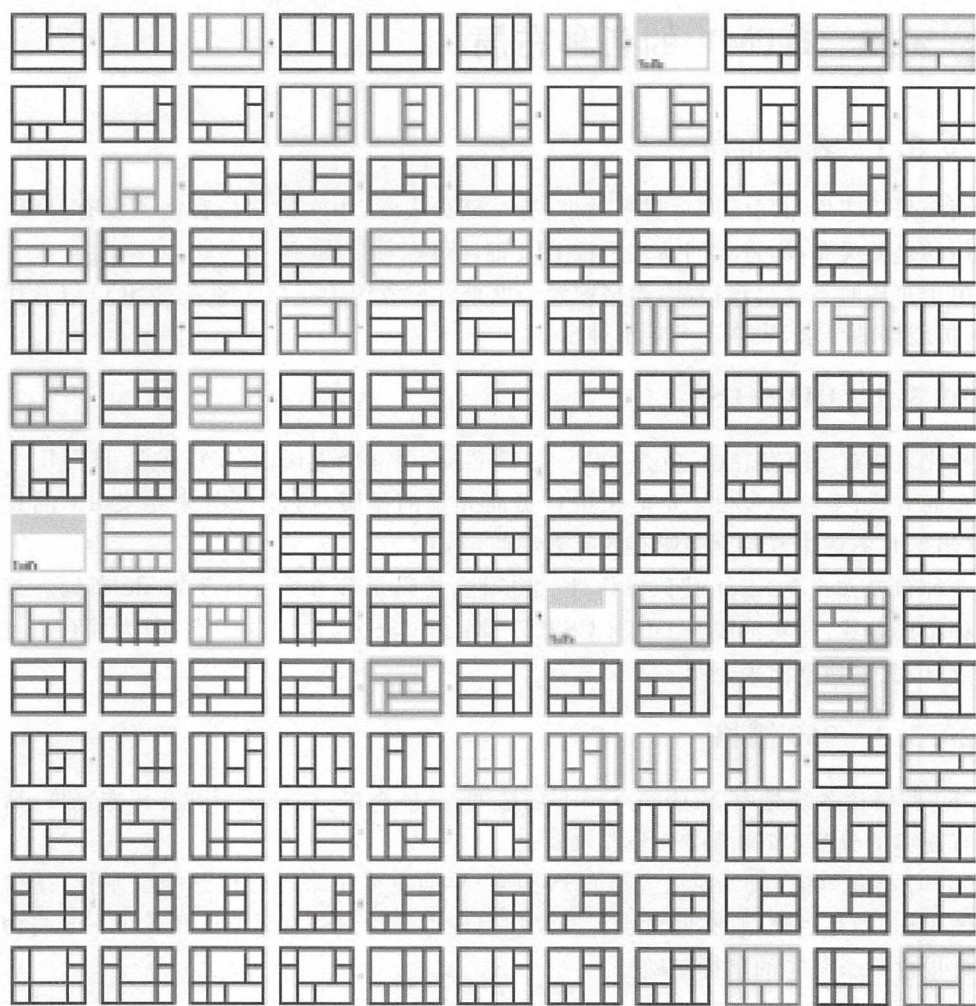


图 20.2 960 栅格

### 20.3.5 瀑布流布局

出于对一些图片类的信息呈现,“蘑菇街”(网站)提出了一种新的布局方式——瀑布流布局。而今,这种布局方式,大量地应用于图片展示效果。它具备着“延时加载图片信息”的优势,随着用户向下滚动页面,不断地请求数据,进行加载。百度当前采用的图片方式也是如此(纵向高度限制,横向自适应,然后进行相关数学运算和细节处理)。瀑布流布局的视觉效果可查看第 7 章。

### 20.3.6 响应式布局

前端开发工程师将已有的开发技巧(弹性网格布局、弹性图片、媒体和媒体查询)整合起来,实现多个终端的兼容,是一种针对任意设备对网页内容进行“完美”布局的显示机制。具体可查看第 15 章内容。





### 20.3.7 单页面无限滚动

2015 年左右,单页面滚动的布局方式开始流行。这种布局当中,整个网站只有一个页面,内容很丰富,不断地向下滚动,可以查看全部的信息。这种网站摒弃了传统的网页展示形式,选择在同一个页面垂直放置所有内容,从而避免了用户需要打开新页面。配合视差滚动的效果,给人很炫目的视觉感受。

但是,不得不说的是,它的适用范围还是比较小的,大量的加载时间算是这种设计趋势发展的一大障碍。

如果是一个小公司的信息页面呈现。这种页面倒是一个不错的选择。这种布局通常会和视差滚动相结合,从而制作出更漂亮的效果。

### 20.3.8 移动端的 rem 自适应布局

移动端发展之后,又出现了自适应布局,即通过 JS 以及 CSS 的控制,让代码在多种分辨率的移动端能够正常呈现,具体详见第 14 章移动端开发的内容。

### 20.3.9 其他布局以及未来

其他耳熟能详的布局,还包括弹性布局(又名 Flex 布局,2016 年 9 月中下旬,微信提出了应用号,其中的一些小程序就采用了弹性布局)、IE 的 Windows 8 网格布局等。

对于布局的发展,通常是基于设计潮流的变化而变化,可以适当关注每年年底的设计潮流解读,了解了设计的变化之后,对于布局未来的改变应该就心里有数了。



## 20.4 编辑器插件安装与应用

每种编辑器都拥有着数百种甚至更多的插件,在此以一个插件为例,讲解插件的安装以及配置,其他所有插件的操作方法与该插件相同。

### 20.4.1 插件安装

#### 1. 插件下载

Emmet 插件下载地址: <http://emmet.io/download/>。

进入页面之后,单击图 20.3 DOWNLOAD 按钮,下载压缩包,之后将压缩包解压。

#### 2. 文件加载完毕之后,调整文件位置

选择 Preferences→Browse Packages(汉化后则是“偏好”→“浏览程序包”),然后把解压后的文件夹“emmet-sublime-master”放置于其中即可。

#### 3. 检测是否安装成功

重启 Sublime 编辑器,稍等片刻,等待插件安装完毕(安装时,Sublime 编辑器的最底部有安装状态的显示)。

新建一个文件,存储为 .html 为后缀的网页文件,之后在编辑器中输入英文叹号(!),之后按下 Tab 键,如果能够生成出基本的 HTML 文件结构(包含 doctype、head 和 body),则



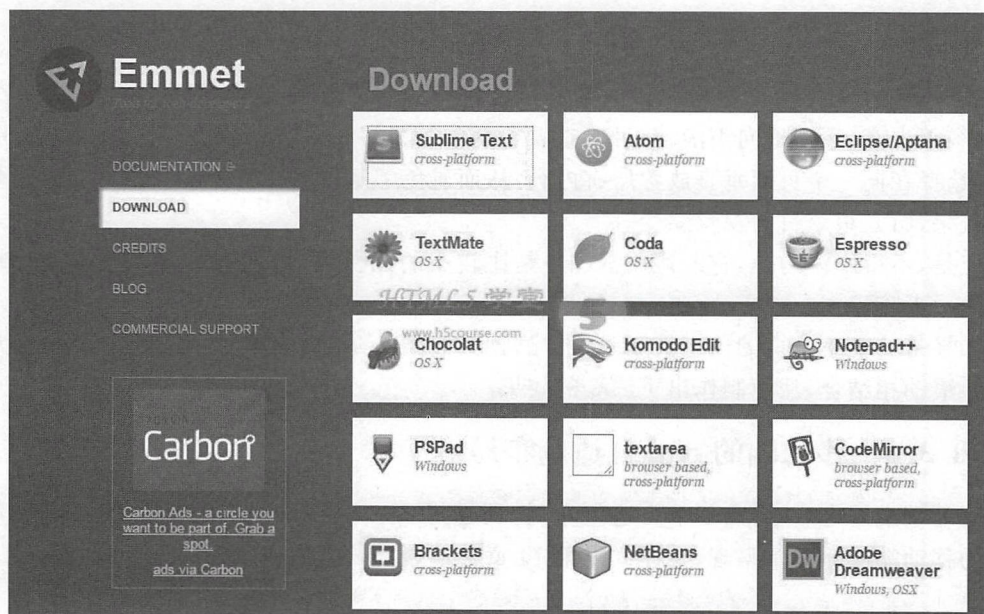


图 20.3 Emmet 插件

说明安装成功。

## 20.4.2 关于 Emmet 插件的相关配置

### 1. 打开配置文件 snippets.json

选择 Preferences→Browse Packages(汉化后则是“偏好”→“浏览程序包”),找到文件夹“emmet-sublime-master”,再在里面找到“emmet”文件夹,之后将其中的 snippets.json,拖曳到 Sublime 当中。

### 2. 修改代码

#### 1) “doc”的修改

按 Ctrl+G 组合键,输入“822”,通过行号快速定位。(不同版本的插件,行数有可能不同,注意修改的是“doc”。)

在此处,将该行修改为如下内容(这一行可以根据需求进行修改,此处只是一个范例)。

代码实例:

```
"doc": "html>(head>meta[charset = UTF - 8] + title{HTML5 布局之路} + link[rel = stylesheet]
[href = ../css/reset.css] + link[rel = stylesheet][href = ../css/index.css]) + body>.wrap",
```

额外备注:

该段代码为一行内容,不能够随意换行或添加空格。格式上不能够有任何错误,否则无法实现最终效果。

#### 2) “html:5”的修改

定位行号 835,将"html:5": "!!! + doc[lang]"修改为"html:5": "!!! + doc"。





### 3. 重启 Sublime,检测安装是否成功

新建一个文件,存储为.html 为后缀的网页文件,之后在编辑器中输入英文叹号(!),之后按下 Tab 键,如果在 Sublime 编辑器中呈现如下效果则说明安装成功。

```
<!doctype html>
<html>
<head>
  <meta charset = "UTF - 8">
  <title>HTML5 布局之路</title>
  <link rel = "stylesheet" href = "../css/reset.css">
  <link rel = "stylesheet" href = "../css/index.css">
</head>
<body>
  <div class = "wrap"></div>
</body>
</html>
```

## 20.4.3 利用 Emmet 插件快速编写 HTML 代码

在安装完成插件之后,想快速书写代码,还有几个必要条件。

### 1. 熟练掌握 CSS 选择器

例如: #con 表示的是选取 id 名为 con 的元素,.wrap 表示的是选取类名为 wrap 的元素,>号表示的是子代选择器,+表示的是毗邻选择器,[ ]则表示属性选择器。

之所以需要熟练掌握选择器的书写方式,主要是由于在快速书写 HTML 代码时使用的方式和 CSS 选择器如出一辙。

### 2. 基础语法

基础语法如表 20.1 所示。

表 20.1 基础语法

基 本 语 法	语 法 含 义
E	元素名称(如: div, p);
E#id	使用 id 的元素(div#content, p#intro, span#error);
E.class	使用类的元素(div.header, p.error.critical);也可以同时设置类名和 ID 名,如: div#content.column
E>N	子代元素(div>p, div#footer>p>span);
E+N	兄弟元素(h1+p, div#header+div#content+div#footer);
E*N	元素倍增(ul#nav>li*5);
\$	表示自动编号
E\$*N	条目编号(ul#nav>li.item-\$*5);
{ }	在大括号当中,用于输入标签的内容
( )	小括号用于提升优先级



### 3. 书写前的构思

这一点是实现代码快速编写最重要的一点。在书写代码前,一定是需要缜密构思的,考虑扩展性,考虑语义性(SEO),考虑标签嵌套的规则,考虑用户交互(是否需要单击,增加a标签),在充分考虑之后再动手。

### 4. 快捷书写实例

代码实例:

```
.wrap>(dl>(dt>a[title = HTML5 布局之路]>img[src = ../images/h5course_$.jpg][alt = HTML5 布局之路 - $]) + dd>(h2>a[HTML5 布局之路 - 文字标题 $ $]) + p[文章 $ $ 的相关描述信息]) * 3
```

按下 Tab 键之后,“奇迹”发生了,代码变成了如下样式。

```
<div class = "wrap">
  <dl>
    <dt><a href = "" title = "HTML5 布局之路"><img src = "../images/h5course_1.jpg" alt =
    "HTML5 布局之路 - 1" title = ""></a></dt>
    <dd>
      <h2><a href = "" title = "">HTML5 布局之路 - 文字标题 01</a></h2>
      <p>文章 01 的相关描述信息</p>
    </dd>
  </dl>
  <dl>
    <dt><a href = "" title = "HTML5 布局之路"><img src = "../images/h5course_2.jpg" alt =
    "HTML5 布局之路 - 2" title = ""></a></dt>
    <dd>
      <h2><a href = "" title = "">HTML5 布局之路 - 文字标题 02</a></h2>
      <p>文章 02 的相关描述信息</p>
    </dd>
  </dl>
  <dl>
    <dt><a href = "" title = "HTML5 布局之路"><img src = "../images/h5course_3.jpg" alt =
    "HTML5 布局之路 - 3" title = ""></a></dt>
    <dd>
      <h2><a href = "" title = "">HTML5 布局之路 - 文字标题 03</a></h2>
      <p>文章 03 的相关描述信息</p>
    </dd>
  </dl>
</div>
```

备注: 在开发时多尝试使用快捷书写,渐渐地就能够掌握这种书写方式。





## 20.5 开发需要准备的基本软件

### 20.5.1 基本软件列表

编辑器：建议使用推荐的 Sublime Text 软件，也可以选用提到的其他各类开发工具。

笔记类：建议使用印象笔记或有道云笔记，能够及时地将自己的学习笔记录下来，并且移动设备能够与 PC 端设备同步，能够使用手机随时查看。

浏览器类：建议安装谷歌浏览器（强烈建议在日常时使用谷歌浏览器）、火狐浏览器、IE9+浏览器、360 安全浏览器（LOGO 是一个绿色的 e）、360 极速浏览器（LOGO 是一个五色花），以上为最基本的浏览器，主要用于代码的调试。

服务器类：建议安装 WAMP 软件（用于将本地机器模拟成服务器），在真正开发当中还需要安装 FTP 工具（用于将文件从本地上传到真正的服务器），可以使用 FlashFxp 软件。

协同开发类：协同开发类软件可以选用 SVN、Git 工具。

图片处理类：建议安装 Photoshop CS5+版本（CS5、CS6 或 CC，如果计算机配置较低，可以选用 CS3 版本），另外建议安装光影魔术手，用于进行一些图片的批量处理和简单操作（没必要使用 PS 处理的图片）。

其他类：建议安装 Microsoft Office Word 和 Adobe Reader 软件，用于查看一些资料信息。

### 20.5.2 WAMP 软件的安装

#### 1. WAMP 是什么？

WAMP 就是服务器集成环境（Windows Apache MySQL PHP 集成安装环境），即在 Windows 下的 Apache、PHP 和 MySQL 的服务器软件。简单来说，就是在自己计算机上搭建了一个虚拟的服务器，而这个服务器里面要用到的后台环境、数据库都会在 WAMP 的安装当中自动完成。

#### 2. WAMP 服务器的安装流程

- （1）弹出安装向导，进行下一步，同意安装协议（I accept the agreement），继续下一步；
- （2）选择安装目录，默认是 C 盘下的 wamp 文件夹，此处可以根据自己需求选择路径，但是请直接放置在某一个盘当中，并且文件夹不要使用中文名；
- （3）之后安装程序会询问是否在快速启动栏和桌面创建快捷方式，也可以不进行设置，直接下一步；
- （4）单击（install）开始安装；
- （5）弹出对话框，询问是否安装新的 WAMP 主页，选择同意后会覆盖 wamp 安装目录下 www 文件夹中的 index.php 文件，可以单击“同意”；

(6) 安装完毕,询问是否启动 WAMP,单击 Finish 按钮完成安装。

3. WAMP 的基本操作

(1) 启动 WAMP: WAMP 安装完毕之后,WAMP 默认为自动启动;如果没有自动启动,可以找到 wamp 文件夹中的 wampmanager.exe,双击运行。

(2) 服务器的启动模式: 启动 WAMP 之后,桌面右下角的任务栏会多出一个 WAMP 的小图标,根据版本不同,小图标样式也不同,较低的 WAMP 版本显示为一个白色的半圆,较高版本的 WAMP 显示的是一个绿色的方框样式。如果半圆不是白色而是红色或黄色,或者方框不是绿色而是红色或黄色,则表示 WAMP 的服务器中某些项(共包括 Apache 和 MySQL 两种服务)启动失败。

(3) 服务器的离线模式: 如果白色图标(较低版本为白色,较高版本为绿色)上面有一个锁,表示的是只能使用本机进行调用(即离线模式)。

(4) 服务器的在线模式: 如果白色图标(较低版本为白色,较高版本为绿色)上面没有任何内容,则说明已经切换到了在线模式,同一个局域网内的计算机或手机都能够通过 IP 地址进行访问。(在图标上单击就可以调出菜单,实现在线、离线模式的切换。)

(5) 切换中文: 右键单击图标,依次选择 Language→ Chinese,即可更换为中文操作界面。

(6) 测试是否能够运行的方法: 打开浏览器,在浏览器窗口当中输入“localhost”,呈现如图 20.4 所示样式即说明安装成功。

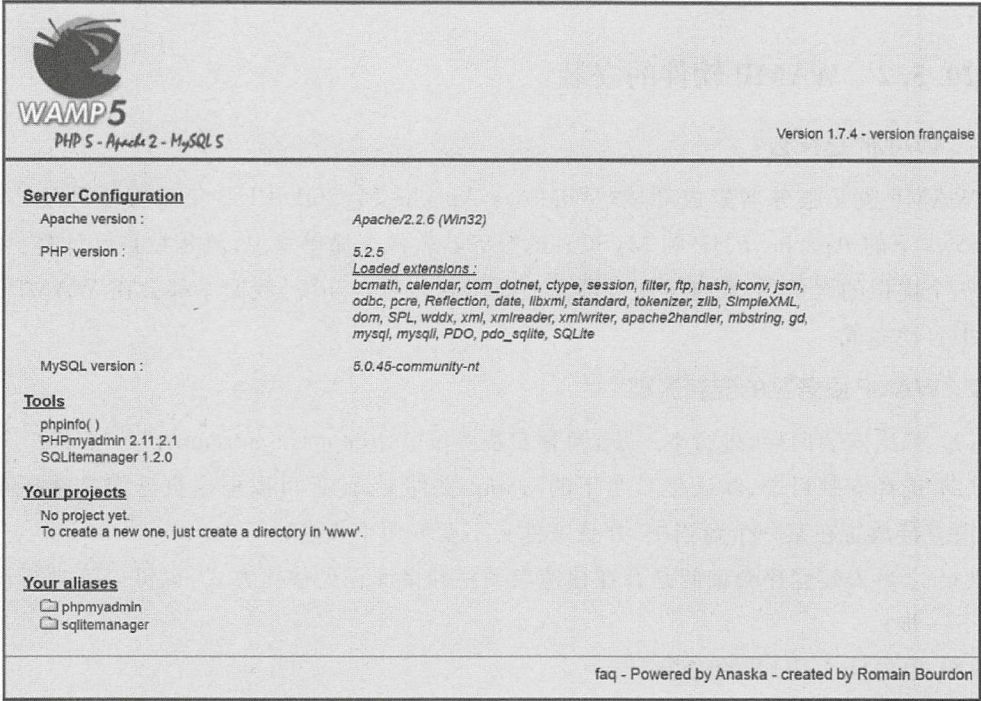


图 20.4 WAMP 预览效果



(7) 代码测试方法：打开 wamp 目录中的 www 目录，如图 20.5 所示，将需要测试的文件夹放置于该目录当中，之后，在 localhost 页面当中就出现了相应的目录，如图 20.6 所示。

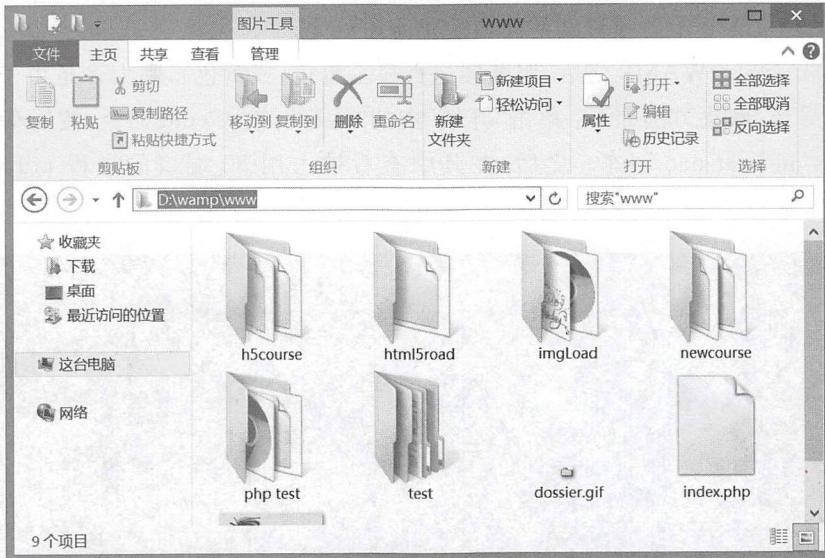


图 20.5 放置文件夹于 www 目录

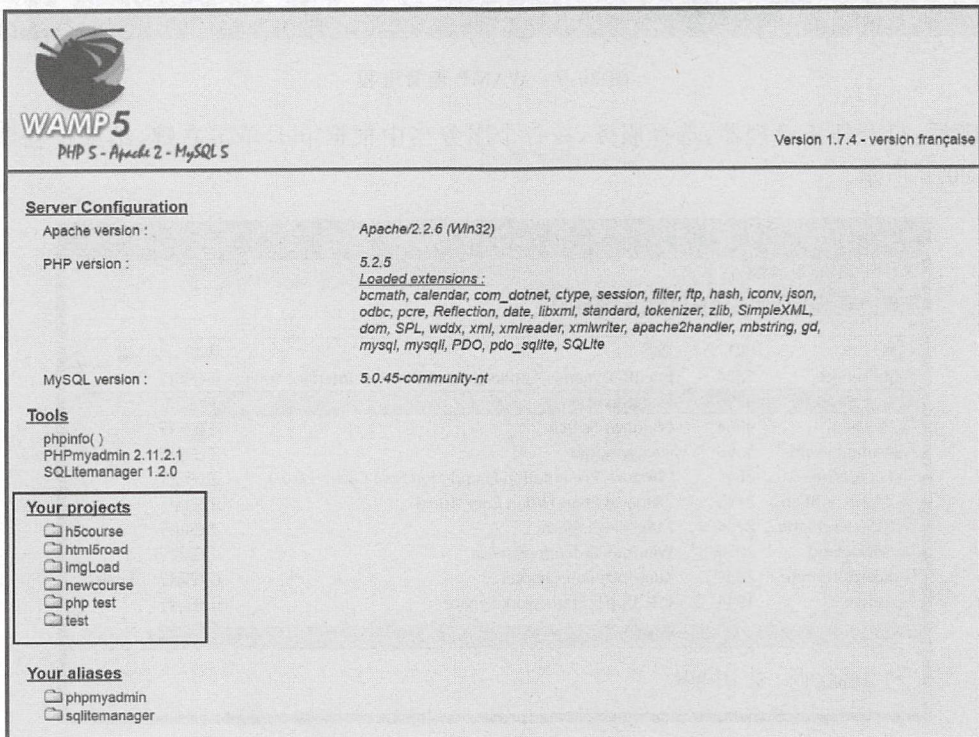


图 20.6 存在文件夹时 WAMP 预览效果

#### 4. WAMP 使用时的注意事项

- (1) 在安装 WAMP 服务器时,必须安装在英文目录下;
- (2) 使用 WAMP 服务器进行页面访问时,需要关闭计算机的防火墙;
- (3) 所有 WAMP 中的项目文件夹均需要使用英文;

(4) 如果安装完 WAMP 之后,服务器运行显示橙色(或黄色),则大多是由于服务器 80 端口被占用导致,解决方法如下:单击“开始”中的“运行”,输入 cmd 命令,按回车键,在命令窗口中输入:netstat-ano 回车;之后,在其中查看被占用 80 端口的进程 pid,如图 20.7 所示。

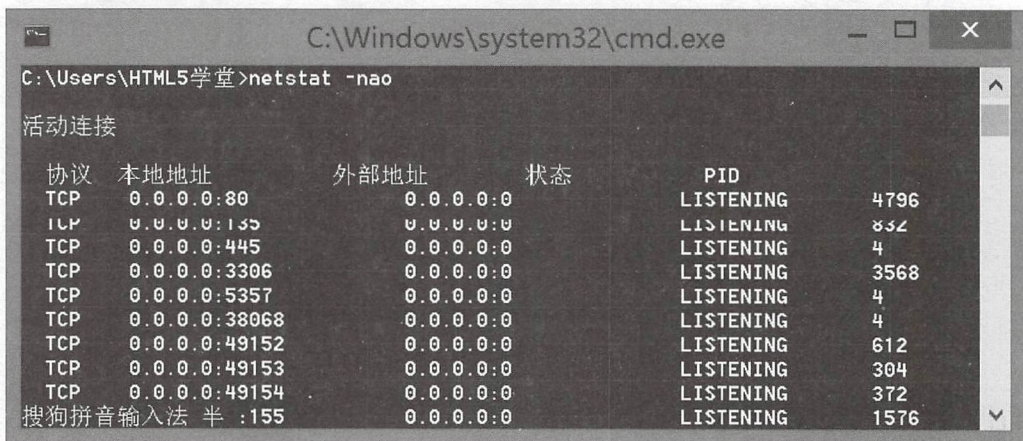


图 20.7 WAMP 查看进程

随后,打开任务管理器,选择服务,在各个服务当中根据 pid,单击右键,结束进程即可,如图 20.8 所示。

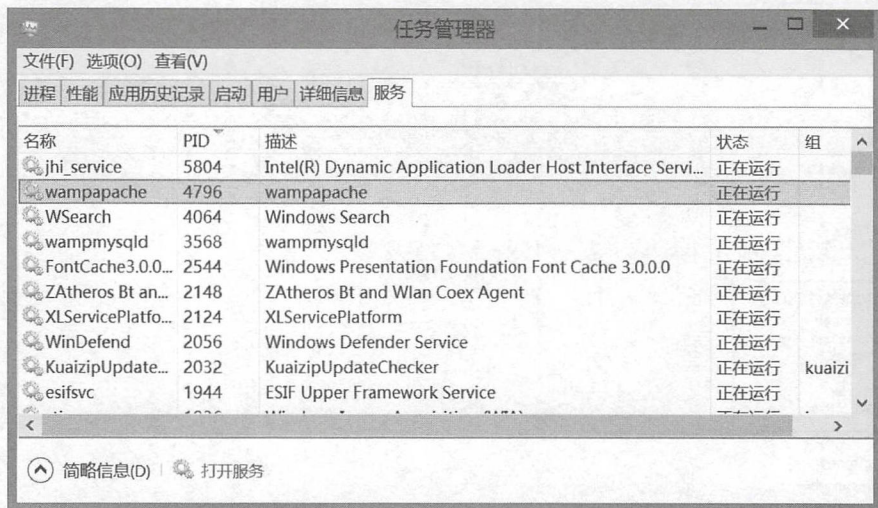


图 20.8 停止占用 80 端口的服务



## 5. WAMP 在实际开发中的用法

(1) 在 PC 端浏览器测试时,网页在 IE9+ 浏览器中进行调试时,需要启动 WAMP 软件,将网页文件放置于 WAMP 目录下的 www 目录当中进行测试,否则外部引入的样式无法加载(WAMP 可以处于离线模式);

(2) 在移动端,进行网页的真机测试时,需要使用到 WAMP,网页文件自然也是放置于 www 目录当中了,此时 WAMP 必须调整为“在线模式”,否则手机无法访问到。

### 20.5.3 初识多人协同开发用的“版本控制管理工具”

#### 1. 版本控制系统

什么是版本控制系统呢?在每天的练习当中,每个人都是独立完成网页的制作与搭建。但是,在实际的开发过程中,较为大型的项目通常需要一个项目组中的多个开发工程师共同完成。此时,大家同时操作一个网站,就有可能出现一系列的问题。比如:A、B 都需要修改页面,于是 A 和 B 把当前的文件下载下来,A 修改页面之后,上传到了服务端,然而 B 并不知情,B 用原来的文件修改之后,也上传到了服务端,此时 B 的文件替换掉 A 的文件,A 的工作就这样白费了。

版本控制系统,就是为了“方便多人协作开发、修改与提交文件、同步变动,在出现冲突问题时快捷解决”而存在的。

#### 2. 较常见的版本控制系统

SVN 和 Git 应该算是当前各个公司开发当中最为常见的两种版本控制系统。SVN 属于“集中式版本控制系统”,而 Git 属于“分布式版本控制系统”。

#### 3. 集中式与分布式版本控制系统的区别

##### 1) 集中式的工作原理

所有的客户端都直接与服务器联系,而彼此不互通。

##### 2) 集中式的特点

集中式的版本控制系统要求必须联网;

在集中式的版本控制系统当中,中央服务器就好比是一个图书馆,如果希望修改一本书,就需要先将书借走,在修改之后,再放回图书馆;

相对安全性会比较低,一旦中央服务器出现问题,所有人都没法干活了。

##### 3) 分布式的工作原理

所有的客户端都可以互相联系,同时任意一个客户端都可以作为一个服务器。

##### 4) 分布式的特点

工作时不需要联网,版本库就在每个人的计算机上;

当多个开发者都修改了同一个文件,那么这几个人只需把各自的修改推送给对方,就可以互相看到对方的修改;

安全性高,每个人计算机里都有完整的版本库,某一个人的计算机坏掉了不要紧,随便

从其他人那里复制一份就能够恢复。

#### 4. Git 的操作流程与学习地址

分为 4 步进行 Git 的学习,分别是:

- (1) 准备工作(软件安装、环境配置等);
- (2) 本地与 GitHub 的仓库创建以及仓库间的“沟通”;
- (3) 理解团队合作方式,并创建“开发”分支;
- (4) 开发用户,从 GitHub 中拉取新分支;在自己本地进行分支操作、处理以及文件方面的操作知识。

图 20.9 和图 20.10 中,左侧为 GitHub 端(服务端),右侧为本地客户端。

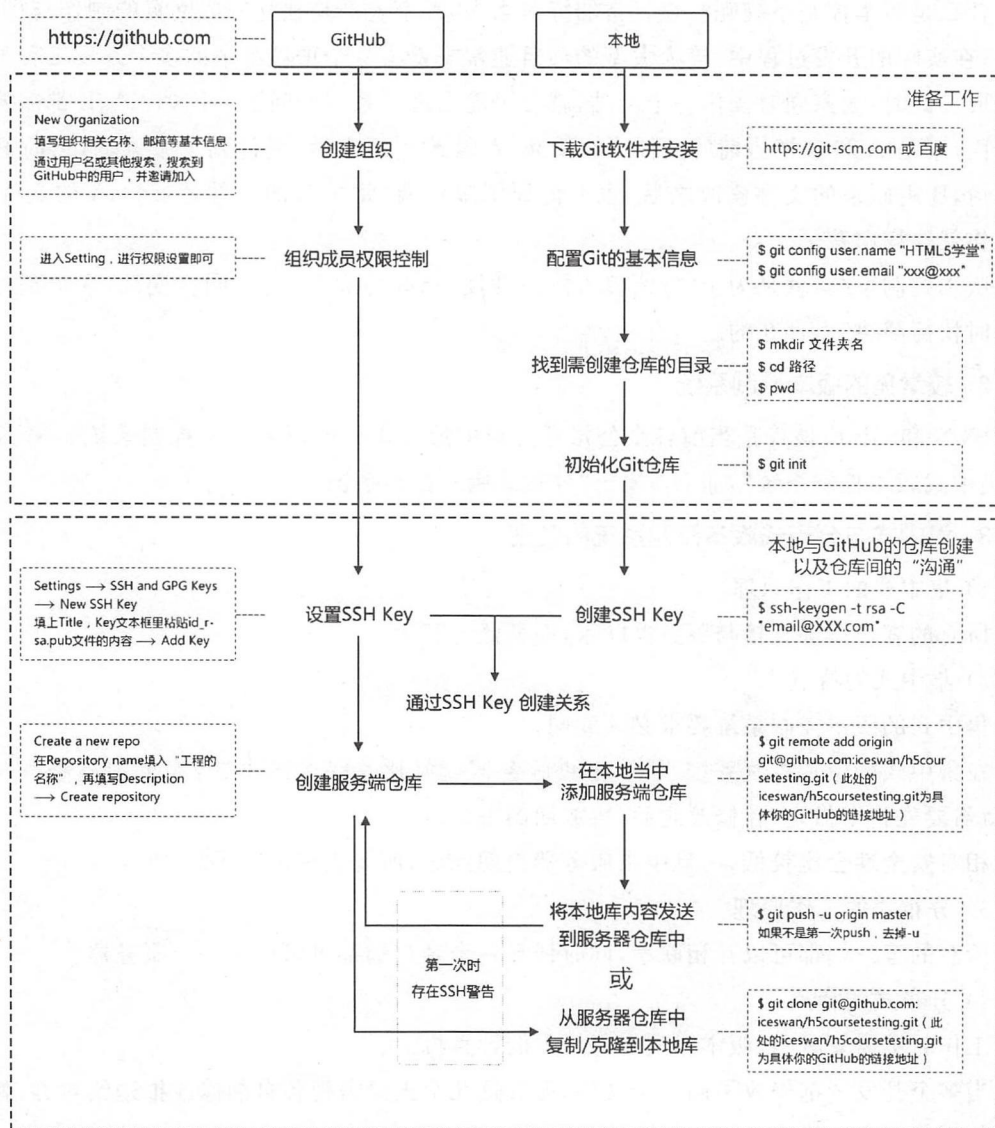


图 20.9 Git 学习流程 1



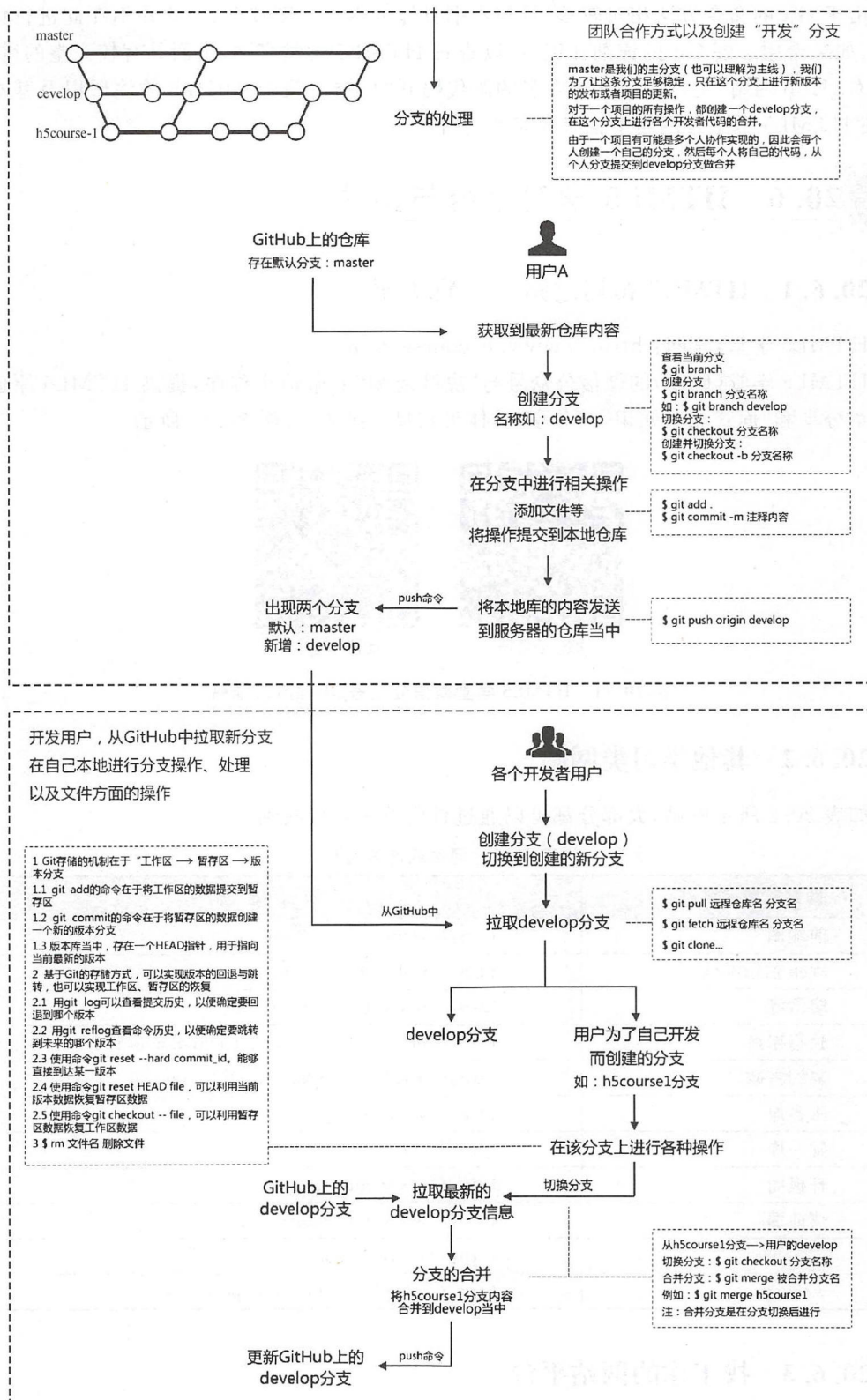


图 20.10 Git 学习流程 2

由于 Git 的命令方法相对较多,且与本书具体内容相关度较低,因此并不在此进行详细讲解,如果希望了解 Git 的详细知识,可以查看 HTML5 学堂官网,里面具有较完整的讲解。对于 GitHub 网站,是行业里比较有名的源代码开源平台,关于 GitHub 的注册以及基本配置,在 HTML5 学堂官网当中也有较完整的讲解。



## 20.6 HTML5 学习平台与网站

### 20.6.1 HTML5 布局之路——官方平台

HTML5 学堂,官网: <http://www.h5course.com>

HTML5 学堂(码匠)的微信公众号与“决胜前端”的微信小程序,提供 HTML5 原创文章分享和“面式相关知识”的分享,具体可扫描二维码,如图 20.11 所示。



图 20.11 HTML5 学堂微信公众号、小程序二维码

### 20.6.2 其他学习类网站

如表 20.2 所示网站,大部分都可以通过百度搜索直接找到。

表 20.2 网站或博客名称

网站或博客名称	网 站 地 址
前端圈	<a href="http://www.fequan.com/">http://www.fequan.com/</a>
Web 伯乐在线	<a href="http://web.jobbole.com/">http://web.jobbole.com/</a>
廖雪峰	<a href="http://www.liaoxuefeng.com/">http://www.liaoxuefeng.com/</a>
独行冰海	<a href="http://blog.163.com/hongshaoguoguo@126/">http://blog.163.com/hongshaoguoguo@126/</a>
梦幻雪冰	<a href="http://ml8050905128.blog.163.com/">http://ml8050905128.blog.163.com/</a>
张鑫旭	<a href="http://www.zhangxinxu.com/">http://www.zhangxinxu.com/</a>
阮一峰	<a href="http://www.ruanyifeng.com/">http://www.ruanyifeng.com/</a>
轩枫阁	<a href="http://www.xuanfengge.com/">http://www.xuanfengge.com/</a>
热前端	<a href="http://www.reqianduan.com/">http://www.reqianduan.com/</a>
w3cplus	<a href="http://www.w3cplus.com/">http://www.w3cplus.com/</a>
w3cfuns	<a href="http://www.w3cfuns.com/portal.php">http://www.w3cfuns.com/portal.php</a>

### 20.6.3 找工作的网站平台

拉勾网、BOSS 直聘、51job、智联招聘、猎聘网、内推网等。







## 20.7 单词列表

HTML5 技术常用单词列表如表 20.3 所示。

表 20.3 HTML5 技术常用单词列表

单 词	含 义	单 词	含 义
style	样式	type	类型
doctype	文档类型	css sprite	背景图压缩工具
charset	字符集	margin	外边距
padding	内边距	border	边框
width	宽	height	高
background	背景	float	浮动
clear	清除	after	后面
before	前面	font-family	字体
SimHei	黑体	SimSun	宋体
Microsoft YaHei	微软雅黑	sans-serif	无衬线体
font-size	字体大小	font-weight	字体粗细
font-style	字体样式	bold	加粗
normal	正常	italic	斜体
text-decoration	文本修饰	underline	下画线
line-height	行高	text-indent	文本缩进
text-align	水平对齐方式	left	左边
right	右边	center	居中
vertical-align	垂直对齐方式	top	上边
bottom	下边	middle	中间
word-spacing	字间距	letter-spacing	字母间距
background-color	背景颜色	background-image	背景图像
background-position	背景定位	background-attachment	定位背景图是否固定
background-repeat	背景重复方式	text-overflow	文本溢出包含元素时发生的事情
clip	裁剪	ellipsis	省略号
nowrap	不进行换行	white-space	处理元素内的空白
color	颜色	overflow	内容溢出时发生的事情
hidden	隐藏		



## 20.8 网页中部分模块的 CSS 命名参考

头: header

内容: content/container

尾: footer

导航: nav

侧栏: sidebar



栏目: column  
页面外围控制整体布局宽度: wrapper / wrap  
左右中: left right center  
登录条: loginbar  
标志: logo  
广告: banner  
页面主体: main  
热点: hot  
新闻: news  
下载: download  
子导航: subnav  
菜单: menu  
子菜单: submenu  
搜索: search  
友情链接: friendlink  
页脚: footer  
版权: copyright  
滚动: scroll  
内容: content  
标签页: tab  
文章列表: list  
提示信息: msg  
小技巧: tips  
栏目标题: title  
加入: joinus  
指南: guild  
服务: service  
注册: register  
状态: status  
投票: vote  
合作伙伴: partner



## 20.9 重置代码解析

### 20.9.1 重置代码

```
1. @charset 'utf-8';  
2. html{color: # 000;background: # FFF;font - family: 'Microsoft YaHei', sans - serif, Arial;}
```





```
3.  body, div, dl, dt, dd, ul, ol, li, h1, h2, h3, h4, h5, h6, pre, code, form, fieldset, legend, input,
    button, textarea, p, blockquote, th, td, strong{padding:0;margin:0;font-family:'Microsoft
    YaHei', sans-serif, Arial;}
4.  table{border-collapse:collapse;border-spacing:0;}
5.  img{border:0;}
6.  a{text-decoration:none;color:#333;outline:none;}
7.  a:hover{text-decoration:underline;}
8.  var,em,strong{font-style:normal;}
9.  em,strong,th,var{font-style:inherit;font-weight:inherit;}
10. li{list-style:none;}
11. caption,th{text-align:left;}
12. h1,h2,h3,h4,h5,h6{font-size:100%;font-weight:normal;}
13. input,button,textarea,select,optgroup,option{font-family:inherit;font-size:inherit;
    font-style:inherit;font-weight:inherit;}
14. input,button,textarea,select{*font-size:100%;}
```

### 20.9.2 重置代码解析

第1行,表示定义CSS文件的字符编码格式为"utf-8"。

第2行,为HTML元素定义如下样式:字体颜色为黑色,背景颜色为白色,字体类型为“微软雅黑、sans-serif、Arial”三种。

第3行,为body、div等一系列元素设置字体类型为“微软雅黑、sans-serif、Arial”三种。

第4行,为table标签设置:合并单元格之间的边框,边框间距为0。

第5行,为img标签设置0像素的边框(img标签在a标签当中时,默认会有蓝色边框)。

第6行,为a标签设置初始样式,没有下画线,字体颜色为#333(深灰),在单击a标签时没有外侧的高亮线(outline)。

第7行,为a标签的hover状态设置样式,出现下画线。

第8行,为var、em、strong元素设置字体样式为“正常”。

第9行,为em、strong、th、var元素设置字体样式为“继承”,字体粗细为“继承”(第8、9行的设置方式,相当于是先将网页中的所有var、em、strong标签的字体样式去除,之后,再设置样式继承父级样式,这样能够保证其他标签中的var、em、strong标签的样式与其父级元素相同)。

第10行,为li去除列表项的小标志。

第11行,为caption、th设置左对齐。

第12行,为h1~h6的标签设置字体大小,均为父级元素的字体大小,且不做加粗处理(默认状态下不同的标题类元素大小不同,且均具有加粗效果)。

第13行,为input、button、textarea等元素设置字体类样式,字体的各种样式均继承父级样式。

第14行,为input、button、textarea、select设置hack,在IE6和IE7下字体为父级的100%。



### 20.9.3 在重置文件中添加的语句

在学习到浮动清浮动的知识时,我们在原有的重置文件当中添加了 after 伪元素清浮动的语句,代码如下。

```
.clearfix:after {
    content: '\200B';
    clear: both;
    display: block;
    height: 0;
}
.clearfix {
    * zoom: 1;
}
```



## 20.10 开发备忘录

良好的项目开发,从缜密的分析与计划开始,充分的项目开发准备能够让之后的问题降低到最少,提升整体开发效率。

### 20.10.1 书写基本的需求分析报告

#### 1. 基于需求分析报告的“任务”

- (1) reset 文件的基本调整;
- (2) 标签的基本选用;
- (3) 典型的布局选用和基本的布局操作;
- (4) 切图工作。

#### 2. 需求分析时需要注意的部分

- (1) 美工图大小和具体内容区域大小的区别,内容区域外的样式如何处理。
- (2) 会不会有 fixed 定位。
- (3) 有没有返回顶部功能需求。
- (4) 哪些地方需要注意超出隐藏。
- (5) 哪些地方需要内容撑开高度。
- (6) 哪些地方需要有链接跳转。
- (7) hover 效果(有些美工图会提供三态:初始、移入和按下)是什么样子。
- (8) 哪些地方需要加光标的小手状态。
- (9) 哪些有 JS 特效,需要考虑效果。
- (10) 基本字体字号、颜色和背景颜色的选用。
- (11) 分清背景图和数据图。
- (12) 公共模块。





(13) 哪些地方是需要提交数据的,考虑使用 form。

## 20.10.2 基本的前期准备工作

- (1) 文件夹结构搭建。
- (2) 准备 reset 重置文件以及基本的几个 JS 文件,根据情况引入。
- (3) 合理修改 reset 文件(基于网站分析结果)。
  - ① 基本的背景颜色和文字颜色。
  - ② 标签的两种状态。
  - ③ 基本的字体大小和样式设置。
  - ④ 删除掉没有使用到的样式设置。
  - ⑤ 保证 after 伪元素清浮动的方法在 reset 当中。

## 20.10.3 移动端与 PC 端的特殊性

### 1. 移动端

- (1) 全新的选择器以及大部分的 CSS3 属性都得到了很好的支持;
- (2) 视口的设置;
- (3) 基本 rem 的处理;
- (4) user-select; tap-highlight-color; -webkit-transform-style 等样式的处理;
- (5) 需要额外注意“指触区”;
- (6) 模拟测试之外,需要进行真机调试;
- (7) 320 屏幕像素下,字体大小最小为 12px;
- (8) a 标签的 title 和 img 标签的 title 可以删除,嵌套原则可以适当调整;
- (9) 最大最小宽的考虑;
- (10) 针对背景图进行 background-size 的处理。

### 2. PC 端

- (1) 使用新标签后,对新标签的兼容处理;
- (2) 需要测试各个浏览器(保证在 IE6、IE7 的低端浏览器中,布局与功能正常);
- (3) 对于子代选择器、CSS3 新增选择器等能否使用取决于具体开发的兼容要求;
- (4) IE 下的测试,需要启动服务器,即在 WAMP 下运行。

## 20.10.4 整体开发的基本顺序提醒

- (1) 注意模块的最小最大宽度/高度;
- (2) 先完成一级布局与二级布局;
- (3) 对于公共部分,统一开发;
- (4) 在完成模块布局之后,再制作具体的模块;
- (5) 边开发边测试。



## 20.10.5 具体开发中的注意事项

### 1. 开发细节

- (1) 文件名禁止使用中文命名；
- (2) 编写代码的时候,需要合理的缩进(不要出现空格与 Tab 混用)与注释；
- (3) 遵循基本的嵌套规则；
- (4) 不要忘记 a 标签的 href 和 title、img 标签的 alt、title、src、a 标签的 target(从何处打开链接)；
- (5) CSS 后代选择器,尽量不要超过 3 层,不要超过 4 层；
- (6) 类名采用单词(语义)命名,多个单词采用中画线连接；
- (7) 不设置不必要的属性和属性值,如针对占满父级整行的块元素设置 width:100%,就是不必要属性；
- (8) CSS 样式按照顺序书写:显示属性→自身属性→文本→其他→CSS3 属性；
- (9) HTML 与 CSS 中的引号需要保持一致,禁止出现单引号与双引号混用；
- (10) 类名和 id 名通常不重复；
- (11) 每段语句结束后的分号(英文)必不可少。

### 2. 防止布局错乱

- (1) 注意保持盒模型大小的一致性(如:增加左右 padding,原有 width 需要变小)；
- (2) 及时清除浮动,并采用合理的清除方式；
- (3) 针对定位元素,处理 z-index 值；
- (4) 数据图需要限制宽高；
- (5) 背景图需要进行合并；
- (6) 对于需要超出隐藏的需求,单行文本隐藏或显示为省略号(……),比如在模块标题需要进行设置,多行文本如果显示区域高度固定,需要设置超出隐藏；
- (7) img 标签需要浮动或设置 display:block,以防止 img 元素下的 3 像素空隙；
- (8) 合理使用群组和后代选择器。

### 3. 代码写完后不可缺少的相关工作

- (1) CSS 压缩；
- (2) JS 压缩；
- (3) 图片压缩；
- (4) ico 文件的制作；
- (5) 404 页面。

## 20.10.6 其他

书写代码的时候,需要综合考虑 SEO、扩展性、代码量以及代码可读性,尽可能寻找最佳解决办法。主动思考很重要。







## 20.11 Iconfont

### 20.11.1 什么是 Iconfont

Iconfont 包含在线图标搜索、图标分检下载、在线存储、矢量格式转换、个人图标库管理及项目图标管理等基础功能,如图 20.12 所示。

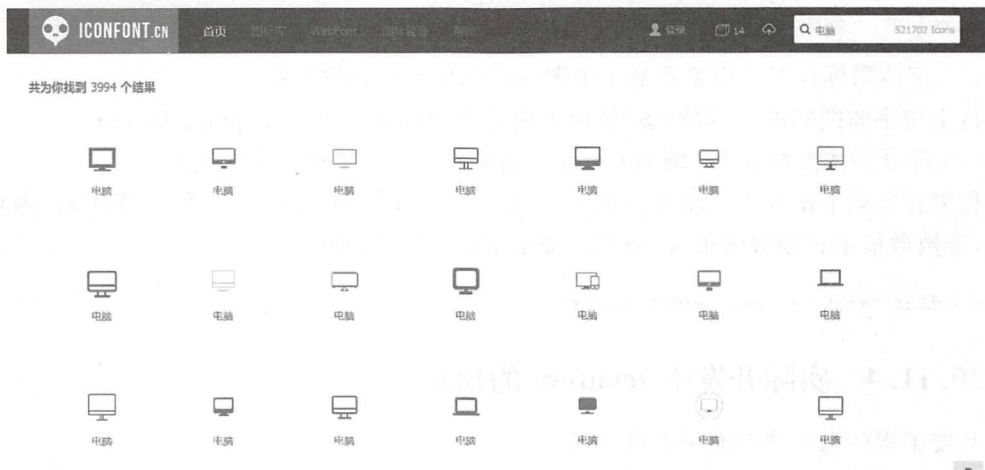


图 20.12 Iconfont 界面

对于 HTML5(前端)开发工程师来说,Iconfont 的主要作用是将一个个 icon 图标转换为特殊字体,从而使网站加载的图片量变小,提升网站的加载速度。

官网地址: <http://www.iconfont.cn/>

### 20.11.2 Iconfont 中图标的下载

在选择需要的图标之后,可以单击“下载至本地”按钮,下载的文件当中,除了具有基本的 icon 文件(以字体样式呈现)之外,还有 CSS 的 demo 文件,如图 20.13 所示。



图 20.13 下载 Iconfont 中的图标



### 20.11.3 Iconfont 可能出现的问题以及解决办法

如下问题以及解决办法是 Iconfont 官网当中所给出的。

(1) 字体图标在 Safari 或 Chrome 浏览器下,有可能被加粗。

由于字体图标存在半个像素的锯齿,在浏览器渲染的时候直接显示一个像素了,导致有背景下的图标显示感觉加粗;所以在应用字体图标的时候需要对图标样式进行抗锯齿处理,CSS 代码设置如下。

```
.iconfont{-webkit-font-smoothing: antialiased;}
```

(2) 字体图标在 IE7 浏览器显示中图标右侧出现小方框现象。

将引用字体图标的“非块标签”转换为块元素即可解决——`display: block;`。

(3) 部分字体图标在 PC 端的 Chrome 浏览器下可能出现严重的锯齿。

如果在谷歌下出现锯齿现象可以对字体图标的边缘进行模糊(只支持 WebKit 内核浏览器,参数数值不宜设置的很大,否则会带来图标加粗的问题)。

```
-webkit-text-stroke-width: 0.2px;
```

### 20.11.4 实际开发中 Iconfont 的用途

开发工程师有可能会面对两种状况:

一种是公司具备靠谱的设计师,进行图像设计,此时可以拜托设计师将设计的 icon 上传到 Iconfont 上,也可以请设计师直接使用 Iconfont 中的图标。之后开发工程师只需要选择相应的 icon 下载即可。

另一种则是公司没有配备设计师,开发工程师在开发页面时还需要自行寻找一些 icon,此时,可以直接使用 Iconfont 提供的各类图标。





## 参 考 文 献

1. 朱印宏. 网页制作与网站开发从入门到精通[M]. 北京:科学出版社, 2009.
2. 咎辉, Zac. SEO 实战密码[M]. 北京:电子工业出版社, 2012.
3. HTML5 学堂[EB/OL]. <http://www.h5course.com>.
4. 刘国利. 独行冰海博客[EB/OL]. <http://blog.163.com/hongshaoguoguo@126>.
5. 陈能堡. 梦幻雪冰博客[EB/OL]. <http://m18050905128.blog.163.com/>.



- 行业知识与开发工具
- 创建HTML文档
- **初识HTML5**

- 文本与背景样式处理
- 定位布局与开发细节
- **文本与细节**

- HTML5与CSS3新知识
- 移动端开发与响应式
- **玩转移动端**

- **标签与布局**
- 网页整体
- 标签选择与模块布局

- **表格表单与复习**
- 表格表单
- PC端知识整理与复习

